Spreading nets: a uniform approach to unfoldings

G. Michele Michele Pinna, Eric Fabre

Spreading nets: a uniform approach to unfoldings

G. Michele Pinna*

Dipartimento di Matematica e Informatica, Università degli Studi di Cagliari, Italy

Eric Fabre

Univ Rennes & INRIA, Rennes, France

Abstract

This paper is devoted to the study of the notion of *spread* net. A spread net is a (safe) Petri nets where each place is annotated with some information, taken from a suitable *information domain*, about how such place can get marked. Spread nets generalize various kinds of nets used to represent the non sequential behaviours of a safe net, like unfoldings, merged processes or trellis processes. The spreading of a net may allow to produce more compact (partially unfolded) nets representing the non sequential behaviour of a net, depending on the chosen information domain. In particular in a spread net it is possible to merge not only conflicting runs, but also to partially refold time, as spread nets allow loops in time.

Keywords: Multi-clock nets, unfolding of nets, spreading of nets

1. Introduction

One of the most popular motto in Petri nets is that "*the semantics of a net is a net*" [1]: the executions of a net N are represented with the aid of another net and a suitable mapping relating the *semantic* net to N. Along this line of thought *non-sequential* processes have been proposed ([2]), where the causal dependencies among the transitions of a net are faithfully represented. Non-sequential processes are able not only to represent causality, but also the concurrency present in a single computation. They cannot however represent conflicts. To model all the possible non sequential executions of a net, the notion of *unfolding* has been proposed in [3] and further investigated in [4] and [5]. The idea is to represent conflicts as branching alternatives (whence the name of *Branching Processes*, that are essentially unfoldings [5]).

Though unfoldings were introduced to represent the non sequential behaviors of (safe) Petri nets, their main application originated in the fact that they offered new techniques for the verification of concurrent systems: the use of partial orders allows one to have more compact representations of a nets behaviors with a gain in how the verification can be performed. Still, the data structure obtained by unfolding 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* ([6] and further investigated in [7]). An orthogonal approach has been the one pursued in [8] where the unfolding is restricted to a finite complete prefix in a way that still allows one to infer all 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 [9] and [10] goes in this direction requiring that each execution is a partial order, but the overall structure gathering all executions does not need to be a partial order. Focussing on the dependencies to obtain a more compact data structure

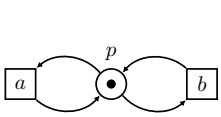
* *Corresponding author.* Dipartimento di Matematica e Informatica, Università degli Studi di Cagliari, via Ospedale 72, 09124 Cagliari (Italy), e-mail: gmpinna@unica.it

implies that the net representing more concisely the non sequential behaviours has to fulfill some properties that are posed on the net as a whole, *e.g.* some subnets should have certain characteristics like the unravel nets, or the net should be acyclic. 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 [11, 12]. 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, hence it is based on the unfolding construction.

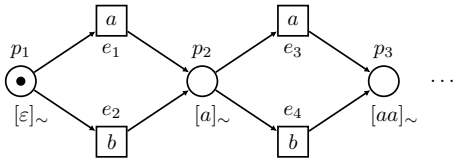
The approaches limiting the size of the data structure having origin in the work regarding prefixes ([8] and further investigated in [13] and [14]) focus mainly on the time dimension. Still the resulting data structure may be unnecessarily big, and in the last decade, merged processes [15] and the closely related trellis processes [16] 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 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.

The problem of representing the non sequential executions of a net, possibly in a more compact way, can be faced from another point of view. Rather than requiring that the *semantic* net representing the non sequential executions of a net obeys to some structural requirements, we define a net where the places are *annotated* with pieces of information. The properties that the data structure representing the behaviours has to fulfill are conveyed through these pieces of information. One of the advantages of this shift in perspective is that it may relay or derive from local update rules, so we do not necessarily have to count on requirements like the existence of a partial order among places and transitions or similar ones.

Spread nets ([17]) are nets where each place is annotated, and the annotation depends on the transitions putting a token in that place and on the annotations of the places feeding these transitions. In this way it is possible to keep track of the way that place is *reached* (obtained) in the executions represented by that net. Based on this notion, we continue the line of research initiated in [17] for multi-clock nets, and we show how the notion of spreading of nets can be seen as a unified approach to Petri net unfolding, also for nets that are not multi-clock nets. 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. Places are annotated by values from an information domain. The notion of information domain associated to a spread net plays a major role. Indeed, the mechanism guiding the *spreading* of a net is encoded in the information domain. We illustrate the intuition with a very simple example, which will allow us to stress how the pieces of information are gathered.



N



N'

Consider the net N above (left), with a single place p and two transitions a and b. The net N' on the left, which the initial part of an unravel net, can be considered the representation of the non sequential behaviour of the net on the left where the various computations are made equivalent provided that each firing of a has to be considered equivalent to each firing of b. This equivalence allows to *merge* the transitions

corresponding to the i-th firing of a or b in the net on the left. The main difference with respect to other approaches is that now the places in the net on the right are *annotated* with a piece of information (in this case it is an equivalence class). As it emerges from this example, the properties one is interested in are *coded* in the information domains, and in this simple example the property one might be interested in could be that the same number of transitions is executed, regardless if it was a or b. Thus, for instance, in N' the place p_2 is annotated with the equivalence class $[a]_\sim$ which contains a and b, or the place p_3 is annotated with the equivalence class $[aa]_\sim$ which contains aa, ab, ba and bb.

This approach extends to more complex ways of representing behaviours thanks to the greater freedom for defining equivalences on the computations. This may allow one to ignore some conflicts but also to refold time (possibly local one), or to abstract only parts of a computation that are considered not relevant. The structure obtained by spreading a net with respect to an information domain can be used to check properties of the system more efficiently, as the spreading may give a more compact structure with respect to previous approaches to represent the executions of a net, and to focus only on events of interest. The annotated places of a reachable marking of a spread net represent the state of the system, and they fully describe all the possible computations leading to that state. All these computations are to be considered equivalent with respect to their common future. The information domain depends mainly on what one would like to consider equivalent in the various activities of the system, *i.e.* indistinguishable from the observations, and this may involve not only conflicts of abstraction but also the folding of executions, which account to say that we can fold time. The semantics of the spread net is reflected in the information domain. The spread net is dependent on it, and in this paper we specify how the domains can be constructed.

We focus mainly on multi-clock nets that are composed of simpler subsystems: basically finite state automata. These automata *synchronize* on common transitions (labels). The resulting net gives us the basic ingredients we want to elaborate on: causality, that coincides with *local time* in each component, and conflicts, which are local to a component as well. Each synchronization among finite state automata determines the expansion of conflicts and causalities to all the components of the system. The information domains for these kind of nets (multi-clock nets) are called *ticking domains* and have a particular structure which depends on the structure of a multi-clock net, and that will be also investigated in this paper.

However the notion of spread net generalizes beyond to multi-clock nets. We introduce it for any kind of net and then we will study how this notion specializes when considering multi-clock nets. The purpose of doing so is to stress that in many case the nets we are interested in may be somehow decomposed in simpler parts, and many times the pieces of information associated to the places in the spreadings may be somehow composed.

This paper is a revised and extended version of [17]. Whereas in [17] only multi-clock nets were considered, here we generalize the notion of spread net to any kind of net. More importantly we show that classical unfoldings can be seen as spread nets, by constructing adequate information domains. This points out the precise relationship among certain information domains and the structure of the resulting spread net. For instance a place of an occurrence net is naturally annotated with the local configuration corresponding to the transition which is in the preset of that place, or in the case of an unravel net, the set of the local configurations. The close relationship among the structure of the spread net is even more evident in the case the unfoldings are originated by a multi-clock net, and we give here a more detailed account on how the information domains are obtained in this case. Furthermore we give more precise criteria on how to construct the ticking domains in the case of multi-clock nets and we discuss various cases of spreading.

Structure of the paper:

The paper is organized as follows. In Section 2 we review Petri nets, and in Section 3 we introduce the notion of *information* domain that will be used to *annotate* nets, obtaining *spread* nets, which are introduced in Section 4. In the Section 4 we also show how various kinds of nets, like 1-occurrence, occurrence and unravel nets can be considered as spread nets over suitable domains, assessing that the notion of spread net is the appropriate one to capture and generalize some of the various unfolding notions presented in literature. In Section 5 we review the notion of *multi-clock* nets (5.1) and, in sub-section 5.2, the notion of *ticking domain*, *i.e.* the domains tailored for multi-clock nets, is introduced and discussed. Section 6 shows that previous unfolding related constructions for multi-clock nets can be seen as spread nets, focussing on

the proper ticking domain characterizing them. In Section 7 we generalize and specialize the notion of ticking domain for a multi-clock net and then we present an algorithm to spread it. We also show that the spreading of a multi-clock net determined by this algorithm is, under some reasonable assumptions, the best construction, in the sense that any other spread net with the same ticking domain is part of the spread net produced by the algorithm. Section 8 illustrates how some useful ticking domains can effectively be constructed. Some conclusions and directions for future research end the paper.

2. Nets

Notation. Given a set A, by $\mathbf{2}^A$ we denote the set of all subsets of A and by $\mathbf{2}^A_{fin}$ the set of *finite* subsets of A. With A^n we denote the product of A n times. With $\mathbb{N}$ we denote the set of natural numbers.

Let A be a set, a *multiset* of A is a function $\mathsf{f} : A \to \mathbb{N}$. The set of multisets of A is denoted by μA. The usual operations and relations on multisets, like multiset union or multiset inclusions, are defined in the standard way. A multiset $\mathsf{f} \in \mu A$ is a set whenever, $\forall a \in A$, $\mathsf{f}(a) \leq 1$, and in the case a multiset f is a set, we often write $a \in \mathsf{f}$ instead of $\mathsf{f}(a) = 1$. Obviously any subset of A can be seen as a multiset of A. Given a multiset $\mathsf{f} \in \mu A$, $[\![\mathsf{f}]\!]$ indicates the set $\{a \in A \mid \mathsf{f}(a) \neq 0\}$ (thus, with abuse of notation, a multiset f is a set when $[\![\mathsf{f}]\!] = \mathsf{f}$). The cardinality of a (multi)set f is denoted by $|\mathsf{f}|$, and it is defined as $\sum_{a \in A} \mathsf{f}(a)$. When no confusion arises we denote the multiset $\mathsf{f} \in \mu A$ as $\sum_{a \in A} \mathsf{f}(a) \cdot a$.

Given a *finite* alphabet A (a *finite* set of symbols), A^* denotes as usual the set of words on A, and ε denotes the empty word. The length of a word $w \in \mathsf{A}^*$ is defined as usual and, with abuse of notation, it is denoted by $|w|$. Given a word $w \in \mathsf{A}^*$ and a subset X of the alphabet, $proj(w, \mathsf{X})$ is the word obtained by deleting all occurrences of symbols not belonging to X. More formally, for a fixed X, $proj(\varepsilon, \mathsf{X}) = \varepsilon$, $proj(a \cdot w, \mathsf{X}) = a \cdot proj(w, \mathsf{X})$ if $a \in \mathsf{X}$ and $proj(a \cdot w, \mathsf{X}) = proj(w, \mathsf{X})$ otherwise. Given a word $w \in \mathsf{A}^*$, $\{\!|w|\!\}$ denotes the set of all symbols of the alphabet A appearing in w, which is defined inductively as $\{\!|\varepsilon|\!\} = \emptyset$ and $\{\!|a \cdot w|\!\} = \{a\} \cup \{\!|w|\!\}$.

Nets. We review the notion of Petri net and of token game and then we recall some further notions we will use in the following.

Definition 2.1. *A* Petri net *is a tuple $N = \langle P, T, F, m_0 \rangle$, 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 and $m_0 \colon P \to \mathbb{N}$ is called the* initial marking.

A net $\langle P, T, F, m \rangle$ 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 $(t, p) \in F$.

Given $x \in T \cup P$, with ${}^\bullet x$ we denote the set $\{y \mid (y, x) \in F\}$ and with $x^\bullet$ the one $\{y \mid (x, y) \in F\}$. ${}^\bullet x$ and $x^\bullet$ are called the *preset* and *postset* respectively of x. Observe that ${}^\bullet x$ and $x^\bullet$ may be considered as multisets (over places or transitions, depending on x) and this will be handy in the following.

A transition t is enabled at a marking m if m *contains* the preset of t, thus if ${}^\bullet t \leq m$. If a transition t is enabled at a marking m it may *fire* yielding a new marking defined by $m' = m + {}^\bullet t - t^\bullet$. The firing of t at m giving m' is denoted with $m \,[t\rangle\, m'$. We will always consider nets where $\forall t \in T$. ${}^\bullet t$ and $t^\bullet$ are not empty. This implies that no transition t can be executed *spontaneously* (${}^\bullet t \neq \emptyset$) and that the execution of a transition has an effect ($t^\bullet \neq \emptyset$), though not necessarily observable (e.g. ${}^\bullet t = t^\bullet$, the firing of t at a given marking cannot be observed only looking at the markings).

Given a marking m, the *firing sequence* (fs) starting at m of the net $N = \langle P, T, F, m_0 \rangle$, is defined as usually:

1. m is a firing sequence (the empty one, where no transition is fired), and
2. if $m \,[t_1\rangle\, m_1 \cdots m_{n-1} \,[t_n\rangle\, m_n$ is a firing sequence and $m_n \,[t\rangle\, m'$ then also $m \,[t_1\rangle\, m_1 \cdots m_{n-1} \,[t_n\rangle\, m_n \,[t\rangle\, m'$ is a firing sequence.

The set of firing sequences of a net N starting at a marking m is denoted by FS^N_m and it is ranged over by σ. We may omit mentioning the superscript denoting the net when it is clear from the context, and the initial marking m when it coincides with the initial marking of the net.

Given $\mathsf{fs}\ \sigma = m\,[t_1\rangle \cdots [t_n\rangle\, m_n$, $start(\sigma)$ denotes the initial marking m and $lead(\sigma)$ denotes the marking m_n, finally $run(\sigma)$ denotes the word $t_1 \dots t_n$. Given a net $N = \langle P, T, F, m_0\rangle$, a marking m is *reachable* iff there exists a $\mathsf{fs}\ \sigma \in \mathsf{FS}^N_{m_0}$ such that $lead(\sigma) = m$. The set of reachable markings of N is $\mathcal{M}_N = \{lead(\sigma) \mid \sigma \in \mathsf{FS}^N_{m_0}\}$. Together with the notion of reachable marking we introduce also the one of *state* of a net which focuses, for each firing sequence, on the executed transitions rather than on the reached marking. Let $\sigma \in \mathsf{FS}^N_m$ be a firing sequence, we define the multiset $\mathsf{st}(\sigma)$ of transitions as $\mathsf{st}(\sigma) = \emptyset$ if $\sigma = m$ and $\mathsf{st}(\sigma) = t + \mathsf{st}(\sigma')$ if $\sigma = m\,[t\rangle\,\sigma'$, and we call it *state* of N. The set of *states* of the net N is the set $\mathsf{St}(N) = \{\mathsf{st}(\sigma) \mid \sigma \in \mathsf{FS}^N_{m_0}\}$.

A net $N = \langle P, T, F, m_0\rangle$ is *acyclic* whenever the transitive and reflexive closure of F is a partial order over $P \cup T$. A net $N = \langle P, T, F, m_0\rangle$ is *live* iff for each transition t and each marking $m \in \mathcal{M}_N$ there exists a firing sequence $\sigma \in \mathsf{FS}^N_m$ such that $\sigma\,[t\rangle$.

We now recall the notion of safe net.

Definition 2.2. *A Petri net $N = \langle P, T, F, m_0\rangle$ is said to be* safe *iff each marking $m \in \mathcal{M}_N$ is a set.*

Though safeness is a *semantics* notion as it is connected with the reachable markings of a net, it can be enforced structurally by adding suitable places (the so called *complementary* places). In this paper we consider safe nets only, and thus we will often identify a marking m with the set $[\![m]\!]$ and write $p \in m$ when $m(p) = 1$ and m will be considered as a set.

Nets morphisms. Let $N = \langle P, T, F, m_0\rangle$ and $N' = \langle P', T', F', m'_0\rangle$ be two safe nets, a *morphism* of safe nets ([4]) is a pair (f_P, f_T) where f_P is a relation between P and P', and f_T is a partial mapping from T to T' such that

1. $f_P(m_0) = m'_0$ and for each $p' \in m'_0$ there exists a unique $p \in m_0$ such that $p' f_P p$,
2. if $f_T(t) = t'$ then the restrictions $f_P^{op} : {}^\bullet t' \to {}^\bullet t$ and $f_P^{op} : t'^\bullet \to t^\bullet$ are total mappings, and
3. if $(p, p') \in f_P$ then the restrictions $f_T : {}^\bullet p \to {}^\bullet p'$ and $f_T : p^\bullet \to p'^\bullet$ are total mappings.

where f_P^{op} denotes the opposite relation of f_P.

A morphism $f : N \to N'$ preserves the initial marking and reachable markings as well, as it maps a firing sequence of N to a firing sequence of N'. When it is clear from the context we may omit the subscript of (f_P, f_T) using f instead.

Labeled nets. Let us fix an alphabet Lab of *labels*.

Definition 2.3. *A* labeled *net over a set of label Lab is a tuple $N = \langle P, T, F, m_0, \ell\rangle$ where*

- *$\langle P, T, F, m_0\rangle$ is a net, and*
- *$\ell : T \to \mathsf{Lab}$ is a total mapping.*

Given a firing sequence $\sigma \in \mathsf{FS}^N_{m_0}$ of the labeled net N, with $\mathsf{tr}(\sigma)$ we denote the word $\ell(run(\sigma))$, often called *trace*. The set of traces of a labeled net N is denoted with $\mathsf{Tr}(N)$. Observe that this set is never empty, as at least the empty word ε belongs to it, which corresponds to the firing sequence m_0. The traces here are not the so called *Mazurkievich-traces* ([18]), as each of them represents the sequence of labels of a single *sequential* execution of the net.

The notion of net morphism is adapted to labeled nets in the obvious way: let (N, ℓ) be a labeled safe net over Lab and (N', ℓ') be a labeled safe net over Lab', a *morphism* is the pair (f, l) where $f : N \to N'$ is a net morphism and $l : \mathsf{Lab} \to \mathsf{Lab}'$ is a partial labeling morphisms such that l is defined on $\ell(T)$ and $\ell'(f_T(t)) = l(\ell(t))$.

As each safe net $N = \langle P, T, F, m_0\rangle$ can be seen as a labeled net on the set of labels T by taking ℓ as the identity mapping, we will add such labeling mapping when needed.

3. Information domains

We introduce now the notion of *information domain* over a set A. This notion is different from the classical notion of Scott domain as defined in the literature (for instance as in [19] and references therein, or [20]), but we prefer to use the term domain as it conveys a similar idea, namely the one that information domains contain pieces of *information* about a system. An information domain will be a non empty set of elements defined from A (*e.g.* subsets of A, or languages in A^*), and we will use the elements of a domain to annotate the places of a net. Each element of the information domain will deliver some information about how the place can be marked, and this piece of information will depend on what one whishes to characterize and observe. We support this idea with some simple examples, which will be formalized later.

Example 3.1. *Let us fix a set* A*.*

- *An information domain can be composed by just one element, say the empty set. This information domain conveys the idea that it is irrelevant to observe or characterize how a given place is marked.*
- *An information domain can be the set of words* A^**. The annotation of a place with a word* w *in* A^* *may characterize the fact that some transitions have been executed to mark that place.* $w \in \mathsf{A}^*$ *may stress that some transitions are executed more than once top produce that place and that the ordering in the execution is relevant.*
- *An information domain can be the set of finite subsets of* A*, namely* $\mathbf{2}^{\mathsf{A}}_{fin}$*. In this case we could consider the annotation of a place as the set of transitions executed, regardless the multiplicity or the ordering.*

The elements of an information domain are in any case related to a predefined set A, and we indicate this fact by saying that the elements are built *over* A.

The elements of an information domain can be *combined*. To combine the elements we define some *partial operations*. The partial operations take pieces of informations (a multiset of elements of the information domain) and return (a multiset of) new pieces of information. The operations are partial as we have have to state when some pieces of information are *coherent*, and in general it is not the case that all the pieces of information can be *combined* together. We define also an *updating* mapping which takes an element of the information domain and adds to it the piece of information delivered by an element in A. The updating mapping is used to calculate new elements starting from an element of the information domain and an element of the set A. Summing up, we equip information domains with two kind of internal operations: one kind to combine elements and the other to update. They are partial operations: they applies to coherent elements.

Formally:

Definition 3.2. *Let* A *be a set, an* information domain $\mathcal{D}$ *(over* A*) is the 4-tuple* $(\mathcal{A}, \mathsf{Op}, \tau, \mathsf{A})$ *where*

1. $\mathcal{A}$ *is a non empty set of elements with a distinguished element denoted with* $\mathsf{0}$*, that will be ranged over by* A*, which we call the* support*,*
2. Op *is a non-empty set of partial operations* $\mathsf{op}^i \colon \mu\mathcal{A} \rightharpoonup \mathbf{2}^{\mathcal{A}}$ *such that, for each* $X \in \mu\mathcal{A}$ *such that* $|X| = i$*, if* $\mathsf{op}^i(X)$ *is defined then* $\mathsf{op}^i(X) \neq \emptyset$*, and*
3. $\tau : \mathcal{A} \times \mathsf{A} \to \mathcal{A}$ *is an* updating *mapping.*

Given A, an information domain $\mathcal{D}$ is basically a partial algebra where A plays the role of the generators of the elements of the algebra. Given $\mathsf{op}^i \in \mathsf{Op}$, i denotes the cardinality of the multiset on $\mathcal{A}$ that the partial operation op^i receives as argument. The partial operation, when defined, combines the i elements and produce a non empty set of elements of $\mathcal{A}$. The operations op^i will be detailed when we introduce the specific information domains, the purpose of each operation is to *combine* i elements in $\mathcal{A}$, that are assumed to be coherent, and produce elements that can be used by the update mapping. Hence the minimal requirement that when it is defined then the result is a non empty subset of elements of $\mathcal{A}$. The set Op is required to be non empty. The final ingredient of an information domain is the *updating* mapping that,

given an element of $\mathcal{A}$ and one of A, produces a new piece of information possibly using the elements in A. We often refer to the support of an information domain as the information domain itself.

We illustrate these operations with some simple examples.

Example 3.3. *Given a set* A*, an information domain over* A *could be composed by setting* $\mathcal{A} = \mathbf{2}^{\mathsf{A}}_{fin}$ *where each element is a finite subset of* A*, with* $\emptyset$ *as the distinguished element* $\mathsf{0}$*. Each operation* $\mathsf{op}^i \in \mathsf{Op}$ *takes a multiset* $X \in \mu\mathbf{2}^{\mathsf{A}}_{fin}$ *of size* i *and returns the union of the elements in* $A \in \mathbf{2}^{\mathsf{A}}_{fin}$ *for which* $X(A) \neq 0$*,* i.e. $\mathsf{op}^i(X) = \bigcup\{A \mid X(A) \neq 0\}$*. The updating mapping just adds one* $a \in \mathsf{A}$ *to an element* $A \in \mathbf{2}^{\mathsf{A}}_{fin}$*,* i.e. *the updating mapping returns the union of the set in* $A \in \mathbf{2}^{\mathsf{A}}_{fin}$ *with the singleton* $\{a\} \in \mathbf{2}^{\mathsf{A}}_{fin}$ *.*

Example 3.4. *Given a finite set* A*, let* $\mathcal{A} = \{\mathcal{L} \mid \mathcal{L} \subseteq \mathsf{A}^*\}$ *be the information domain where each element is a* language $\mathcal{L} \subseteq \mathsf{A}^*$*. The* $\mathsf{0}$ *could be the language* $\{\varepsilon\}$ *(it is not the unique choice, for instance the* $\mathsf{0}$ *could also be* A^**). Each operation* $\mathsf{op}^i \in \mathsf{Op}$ *takes a multiset of languages* X *of size* i *and returns a new set of languages, where each of them is obtained combining the various languages in* X *by some language operations. For instance, a partial operation* op^i *can take a set of languages* $\{\mathcal{L}_1, \ldots, \mathcal{L}_i\}$ *and return the set of languages* $\{\mathcal{L}_k \cap \mathcal{L}_l \mid k \neq l\}$*, or a set of languages* $\{\mathcal{L}_1, \ldots, \mathcal{L}_i\}$ *such that* $\forall k, l \in \{1, \ldots, i\}$ *either* $\mathcal{L}_k \subseteq \mathcal{L}_l$ *or* $\mathcal{L}_l \subseteq \mathcal{L}_k$*, and return the maximal one. For the updating mapping, we can consider the mapping that takes* $\mathcal{L} \in \mathcal{A}$ *and* $a \in \mathsf{A}$ *and returns the language* $\mathcal{L} \cdot a$*.*

As it is, the notion of information domain is rather abstract. We will show how to construct suitable information domains that should convey the idea that places of a net representing the behaviour of a system can be enriched with information that characterizes precisely that behaviour. In some cases the elements of (the support of) an information domain will turn out to be a partial order, but the unique requirement that we make is that it is non empty, as it should contain the $\mathsf{0}$, representing the minimal amount of information available.

We are especially interested in information domains where each operation in Op satisfies some further constraint.

Definition 3.5. *Let* $\mathcal{D} = (\mathcal{A}, \mathsf{Op}, \tau, \mathsf{A})$ *be an information domain over* A*. We say that* $\mathcal{D}$ *is* regular *whenever for each* $\mathsf{op}^i \in \mathsf{Op}$ *and for each* $X \in \mu\mathcal{A}$*, if* $\mathsf{op}^i(X)$ *is defined then* $\mathsf{op}^i(X)$ *is a singleton.*

An information domain is regular iff each partial operation op^i can be seen as partial mapping $\mathsf{op}^i : \mu\mathcal{A} \rightharpoonup \mathcal{A}$.

The information domain in Example 3.3 is regular as each operation returns just one element. This is not necessarily true in Example 3.4.

Example 3.6. *Given* A*, let* $\mathcal{A} = \{\mathsf{0}\}$*.* $\mathcal{A}$ *contains just one element, the* $\mathsf{0}$*. Then* $\mathcal{D}_\perp = (\mathcal{A}, \mathsf{Op}, \tau, \mathsf{Lab})$ *where* $\mathcal{A} = \{\mathsf{0}\}$*,* $\mathsf{Op} = \{\mathsf{op}^i : \mu\mathcal{A} \rightharpoonup \mathcal{A}\}$ *such that* $\mathsf{op}^i(X) = \mathsf{0}$ *for* $X(\mathsf{0}) = i$*, and* $\tau(\mathsf{0}, a) = \mathsf{0}$ *for each* $a \in \mathsf{A}$ *is an information domain over* A*. The* $\mathcal{D}_\perp$ *is regular.*

Example 3.7. *Let* $\mathsf{A} = \{\mathsf{0}, \alpha, \beta\}$ *and let* $\mathsf{op}^1(\mathsf{0}) = \{\alpha\}$*,* $\mathsf{op}^1(\alpha) = \{\alpha, \beta\}$ *and* $\mathsf{op}^1(\beta) = \{\alpha\}$*. This information domain, regardless of the updating mapping, is not regular, as* $\mathsf{op}^1(\alpha)$ *returns a set which is not a singleton.*

Regularity is not the unique interesting *property* of an information domain, we would have also information domains that are *revealing*, where revealing means that the update mapping produces a *new* element, and the pieces of information are updated.

Definition 3.8. *Let* $\mathcal{D} = (\mathcal{A}, \mathsf{Op}, \tau, \mathsf{A})$ *be an information domain over* A*. We say that* $\mathcal{D}$ *is* revealing *whenever* $\mathcal{A} \neq \{\mathsf{0}\}$ *and there exists an element* $A \in \mathcal{A}$ *and an element* $a \in \mathsf{A}$ *such that* $\tau(A, a) \neq A$*.*

A revealing domain must have a support containing also elements which are not $\mathsf{0}$ and these elements can be somehow *obtained* as the result of the update mapping. The information domain $\mathcal{D}_\perp$ is not revealing, whereas the one in Example 3.3 is a revealing one, as we will see in the next section when discussing occurrence nets as spread nets.

4. Spread nets

In this section we review the notion of *spread* net and we show that it may be used to describe the true concurrency semantics of a net, as we pointed out in the introduction. The semantics of a Petri net N has been often given in term of another suitable net (following the "motto" *the semantics of a net is a net*, [1]). Each of the various semantics has relied on an appropriate subclass of nets (*e.g* [3, 21, 2, 22] and [23] among many others), and these *semantics* nets usually defined by some structural characterizations, *e.g.* they have to be acyclic, or each transition has to fire just once.

The idea behind the notion of spread net is rather simple: the places of the net are annotated with information from a suitable information domain $\mathcal{D}$, and the annotation of each place in the postset of some transition t is related to the annotations of the places in the preset of t using the partial operations op^i and the mapping τ. These annotations fully characterize the spread net, as we will see later. Given a multiset $\mathsf{f} \in \mu\mathsf{A}$ and a mapping $g : \mathsf{A} \to \mathsf{B}$, with $g(\mathsf{f})$ we denote the multiset in $\mu\mathsf{B}$ defined as $\sum_{a \in \mathsf{A}} \mathsf{f}(a) \cdot g(a)$.

Definition 4.1. *Let $N = \langle P, T, F, m_0, \ell \rangle$ be a labeled safe net over Lab, let $\mathcal{D} = (\mathcal{A}, \mathsf{Op}, \tau, \mathsf{Lab})$ be an information domain over Lab with $0 \in \mathcal{A}$ as distinguished element, and let $h \colon P \to \mathcal{A}$ be a total mapping, called the* information *mapping, such that*

1. *for each $p \in m_0$ $h(p) = 0$,*
2. *for each $t \in T$, $\mathsf{op}^i(h({}^\bullet t))$ is defined, with $i = |{}^\bullet t|$,*
3. *for each $p \in t^\bullet$, there exists $A \in \mathsf{op}^i(h({}^\bullet t))$ such that $h(p) = \tau(A, \ell(t))$, and*
4. *$\forall p \in P$, $\forall t, t' \in {}^\bullet p$. $\tau(\mathsf{op}^i(h({}^\bullet t)), \ell(t)) = \tau(\mathsf{op}^i(h({}^\bullet t')), \ell(t'))$.*

Assume that for each $t \in T$ there exists a $\sigma \in \mathsf{FS}^N$ such that $\sigma\,[t\rangle$. Then N is a spread *net with respect to the information domain $\mathcal{D}$ over Lab with the information mapping h, and it will be denoted by $N_{\mathcal{D},h}$.*

The new ingredient of a spread net, which is a safe net where each transition can be fired, is the information mapping h. The requirements we put on this mapping are justified as follows:

- condition 1 assures that the piece of information $h(p)$ associated to the initial places is minimal (0), as initial places represent the beginning of a computation and then no information is available,
- conditions 2 and 3 state that the annotation of a place p in the postset of a transition t is obtained by the annotations in the preset t, combined using a partial operation that should be defined on these inputs, using the label $\ell(t)$ of the transition t, conveying the idea that this execution of the transition t adds some information that is gathered in the places in $t^\bullet$, and
- condition 4 ensures that the annotation associated to the place p does not depend on a specific transition in its preset but it is *uniform* over the various transitions in ${}^\bullet p$,

For the time being we do not specify the choice of a specific element from $\mathsf{op}^i(h({}^\bullet t))$ in condition 3. Observe that if the information domain is a regular one then the problem of selecting a specific element does not apply. As a consequence of condition 2, op^i is assumed to be defined for the set of annotations related to the places in the preset of each transition of the net.

We illustrate the idea with a small example, whose purpose is to show how a net can be annotated over an information domain.

Example 4.2. *Consider the information domain $\mathcal{D} = (\mathcal{A}, \mathsf{Op}, \tau, \mathsf{Lab})$, with $\mathsf{Lab} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}\}$, where $\mathcal{A} = \{0, \alpha, \beta, \gamma\}$, (with the elements defined over Lab, for instance $0 = \emptyset$, $\alpha = \{\mathsf{a}\}$, $\beta = \{\mathsf{b}\}$, $\gamma = \{\mathsf{c}\}$ and $\delta = \{\mathsf{d}\}$), Op contains op^1 defined as $\mathsf{op}^1(0) = \{0, \alpha\}$ and $\mathsf{op}^1(\gamma) = \{0\}$, and op^2 defined as $\mathsf{op}^2(2\beta) = \{\beta\}$ and $\mathsf{op}^2(0{+}\alpha) = \{0\}$, and τ is defined as $\tau(0, \mathsf{a}) = 0$, $\tau(\alpha, \mathsf{a}) = \alpha$, $\tau(0, \mathsf{b}) = \beta = \tau(\alpha, \mathsf{b})$, $\tau(\alpha, \mathsf{c}) = \tau(\beta, \mathsf{d}) = \gamma$ and $\tau(0, \mathsf{e}) = 0$. The net in Figure 1 is a spread net with respect this information domain with the following information mapping $h(p_0) = 0$, $h(p_1) = 0$, $h(p_2) = \alpha$, $h(p_3) = h(p_4) = \beta$ and $h(p_5) = \gamma$.*

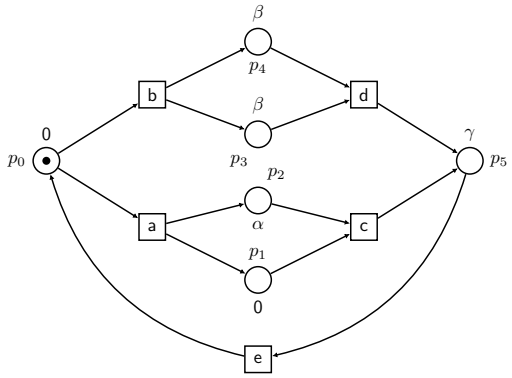


Figure 1: A spread net

Each net can be seen as a spread net over the information domain $(\{0\}, \mathsf{Op}, \tau, \mathsf{Lab})$ where τ is the mapping returning just 0 and each mapping in Op returns the set containing the unique element of $\mathcal{A}$.

Proposition 4.3. *Let $N = \langle P, T, F, m_0, \ell \rangle$ be a labeled safe net over the set of labels Lab, let $\mathcal{D}_\perp$ be the information domain over Lab where $\mathcal{A} = \{0\}$, Op and τ are defined as mappings returning always 0, and let $h : P \to \mathcal{A}$ be the information mapping defined as $h(p) = 0$ for each $p \in P$. Then $N_{\mathcal{D}_\perp,h}$ is a spread net over $\mathcal{D}_\perp$.*

Proof. $\mathcal{D}_\perp$ is a regular domain (see Example 3.6) as for each i we have that $\mathsf{op}^i(X) = 0$ with $X(0) = i$ and $\tau(0, a) = 0$ for each $a \in \mathsf{Lab}$. The conditions of Definition 4.1 are easily verified: to each place in the initial marking the 0 information is associated, $\forall p \in P$ such that ${}^\bullet p \neq \emptyset$ (*i.e.* $p \in {}^\bullet t$ for some $t \in T$), we have that $\forall t \in {}^\bullet p$, $\mathsf{op}^i(h({}^\bullet t))$ is defined and the result is 0, and finally $\tau(0, \ell(t)) = 0$. □

This proposition does not contrast the intuition that a spread net should somehow represent a kind of a net semantics, as it simply states that also a trivial semantics is always available, namely the one where no *information* is provided about the past of a place, hence the net representing the possible behaviours of another one is just the same net.

Definition 4.4. *The spread net $N_{\mathcal{D},h}$ over the information domain $\mathcal{D}$ is said to be* properly *spread whenever for some transition $t \in T$ there exists a place $p \in {}^\bullet t$ and a place $p' \in t^\bullet$ such that $h(p) \neq h(p')$.*

A spread net is properly spread whenever the annotations of some places in the preset and postset of some transition are different. The intuition behind this notion is again obvious: when the annotations do not change, the *spreading* itself (the annotations on the places of the net) is not revealing.

Definition 4.5. *The spread net $N_{\mathcal{D},h}$ is said to be* uniformly *spread whenever $\mathcal{D}$ is a regular domain.*

4.1. 1-Occurrence, occurrence and unravel nets seen as spread nets

To give ground to the notion of spread net, we review briefly some subclasses of nets (among many possible) that have been proposed as the semantics of nets and we see them as suitable spread nets. For each of them we introduce the information domain that establish this connection. The first one captures the idea that in the execution keeping track of dependencies or alternatives is not relevant, thus many computations may be seen as equivalent with respect to dependencies and conflicts; the second captures the opposite idea: dependencies and alternative are to be preserved faithfully, while the third one allows to equate some computations.

The elements of the information domain represent the pieces of information that can be associated to a place p of the net N, namely the piece of information associated to a computation of the net marking the place p. In the remaining of this section, we concentrate on the *semantics* nets of N.

1-Occurrence nets. A 1-occurrence net, as introduced in [24], is a (safe) net where each transition t may fire just once.

Definition 4.6. *A safe Petri net $N = \langle P, T, F, m_0 \rangle$ is said an* 1-occurrence *net if each state in $\mathsf{St}(N)$ is a set.*

In an 1-occurrence net all the different ways a place p receives a token have to be considered as equivalent. A suitable information domain is the one where $\mathcal{A}$ has just one element, namely the $\mathsf{0}$. A labeled 1-occurrence net $N = \langle P, T, F, m_0, \ell \rangle$ where ℓ is the identity may be seen as a spread net over the information domain $\mathcal{D}_\perp$ over $\mathsf{Lab} = T$ stating that the piece of information attached to each place is $\mathsf{0}$.

Proposition 4.7. *Let $N = \langle P, T, F, m_0, \ell \rangle$ be an 1-occurrence net where ℓ is the identity mapping, and let $\mathcal{D}^T_\perp = (\mathcal{A}, \mathsf{Op}, \tau, T)$ be the information domain where*

- *$\mathcal{A} = \{\mathsf{0}\}$, with $\mathsf{0} = \emptyset$,*
- *Op is such that each $\mathsf{op}^i \in \mathsf{Op}$ returns always $\mathsf{0}$ for all $X \in \mu\mathcal{A}$ with $|X| = i = |{}^\bullet t|$ for some $t \in T$, and*
- *$h \colon P \to \mathcal{A}$ associates to each place $p \in P$ the element $\mathsf{0} \in \mathcal{A}$,* i.e. *$\forall p \in P.\ h(p) = \mathsf{0}$.*

Then $N_{\mathcal{D}^T_\perp, h}$ is a spread net over $\mathcal{D}^T_\perp$.

Proof. See Proposition 4.3. □

Observe that the 1-occurrence net is not properly spread, and indeed the presence of the information domain does not add any particular value to the main characteristic of the net, namely the fact that each transition is executed just once. The information domain is regular but it is not revealing.

Example 4.8. *Consider the 1-occurrence net in Figure 2. In this net each place is annotated with* $\mathsf{0}$.

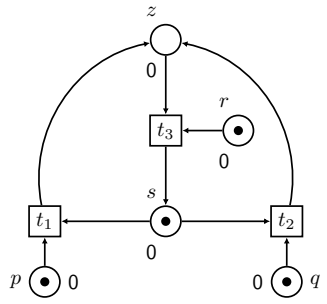


Figure 2: An 1-occurrence net annotated with an information mapping on $\mathcal{D}_\perp$

We stress here that being an 1-occurrence net does not depend on the chosen information domain, but this information domain is in general the one tailored for this kind of net, where all the computations are considered as equivalent.

Occurrence nets. An *occurrence net* [3, 4] is a safe net where dependencies and conflicts among transition occurences, called *events*, can be inferred from the structure of the net itself. The places of an occurrence net are called *conditions* and its set is denoted as B, whereas the transitions are called *events* and its set is denoted as E.

Definition 4.9. *An* occurrence net $C = \langle B, E, W, c_0 \rangle$ is an acyclic safe net such that satisfying the following restrictions:

1. $\forall b \in c_0,\ {}^\bullet b = \emptyset$,

2. $\forall b \in B.\ \exists b' \in c_0$ such that $b'\ W^*\ b$,
3. $\forall b \in B.\ |{}^\bullet b| \leq 1$ and ${}^\bullet b = \emptyset$ implies that $b \in c_0$,
4. $\forall e \in E$ the set $\{e' \mid e'\ W^*\ e\}$ is a finite set, and
5. the relation $\#_C \subseteq E \times E$ defined as
   - $e\ \#_i\ e'$ iff $e \neq e'$ and ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$, and
   - $x\ \#_C\ x'$ iff $\exists y, y' \in E$ such that $y\ \#_i\ y'$, $y\ W^*\ x$ and $y'\ W^*\ x'$,

   is an irreflexive relation.

In an occurrence net each condition (place) not in the initial marking is *produced* by just one event (transition), whereas the ones in the initial marking are produced by none ($\forall b \in c_0,\ {}^\bullet b = \emptyset$). Being C an acyclic net, the relation $\leq_C$ defined as W^* is a partial order on $B \cup E$, and it is often called the *causality* relation. Furthermore each event has a finite number of predecessors, and the conflict relation induced by the forward conflicts among events and inherited along the causality relation. The two relations of causality $\leq_C$ and conflict $\#_C$, are able to characterize the computations of an occurrence net, as each state of the occurrence net is a subset causally closed (to the left) subset of events $X \subseteq E$ which is *conflict free*, *i.e.* $\forall e, e' \in X.\ \neg(e \#_C e')$ and if an event e belongs to X then also the events preceding this one in the partial order are in X as well. Finally we notice that $\leq_C$ is a well founded partial order, and $\forall e \in E$, the finite set $\{e' \in E \mid e' \leq_C e\}$ is denoted with $\lfloor e \rfloor$.

An occurrence net can be seen as a spread net. The piece of information associated to each condition (place) of the net is basically the set of events (transitions) that have been executed in order to *produce* this condition. Indeed in an occurrence net each condition b has a unique history which is easily obtainable taking all the events $\{e \mid e \leq b\}$.

Summing up, the occurrence net $C = \langle B, E, W, c_0, \ell \rangle$ with ℓ being the identity, can be seen as a spread net over the regular domain $\mathcal{D} = (\mathcal{A}, \mathsf{Op}, \tau, E)$ with $\mathcal{A} = \mathbf{2}^E_{fin}$, the operations $\mathsf{op}^i \in \mathsf{Op}$ returning just the union of i places labels, and $\tau(A, e)$ just adding the event in $e \in E$ to the subset of events A.

Proposition 4.10. *Let $C = \langle B, E, W, c_0, \ell \rangle$ be a labeled occurrence net where ℓ is the identity, let $\mathcal{D} = (\mathcal{A}, \mathsf{Op}, \tau, E)$ be the domain where $\mathcal{A} = \mathbf{2}^E_{fin}$, with $\mathsf{0} = \emptyset$, each $\mathsf{op}^i \in \mathsf{Op}$ is defined as $\mathsf{op}^i(X) = \bigcup\{A \mid X(A) \neq 0\}$ and $\tau(A, e) = A \cup \{e\}$, and let the information mapping $h \colon B \to \mathcal{A}$ defined as $h(b) = \{e \in E \mid e \leq_C b\}$. Then $C_{\mathcal{D},h}$ is a spread net over $\mathcal{D}$.*

Proof. First of all we observe that $\mathcal{D}$ is an information domain. Each condition b in the initial marking is annotated with the empty set which is the $\mathsf{0}$ of $\mathbf{2}^E_{fin}$, as ${}^\bullet b = \emptyset$, being C an occurrence net. Consider now a condition b not in the initial marking, then ${}^\bullet b = \{e\}$ for some $e \in E$. Consider the conditions in the preset of e. To each condition $b' \in {}^\bullet e$ is associated a subset of events, on these subsets the union is defined and $h(b) = \{e\} \cup (\bigcup_{b' \in {}^\bullet e} h(b'))$ and $\bigcup_{b' \in {}^\bullet e} h(b')$ is precisely $\mathsf{op}^i(X)$ where X is the multiset obtained by $h({}^\bullet e)$. $\tau(X, t)$ just add the event e to X. The last condition of Definition 4.1 is trivially satisfied as for each condition b we have that ${}^\bullet b$ is at most a singleton. Thus $C_{\mathcal{D},h}$ is a spread net over $\mathcal{D}$. □

Observe that $C_{\mathcal{D},h}$ is properly spread as each place is labeled by its unique history and that $\mathcal{D}$ is regular, furthermore the devised domain is a revealing one.

Example 4.11. *Consider the occurrence net in Figure 3. The annotations of conditions are rather obvious, namely they reflect the set of events that come before (in the partial order) each condition. The $\mathsf{0}$ of the information domain is obviously the empty set and annotates the conditions in the initial marking. For what concerns the operations on the information domain $\mathsf{op}^i(X_1, \ldots X_i)$ is $\bigcup_{1 \leq k \leq i} X_k$ and $\tau_b(X, e) = X \cup \{e\}$.*

It is a well known result that to each occurrence net C it is possible to associate a prime event structure ([4]). A prime event structure is the triple $(E, \leq, \#)$ where E is a set of *events*, $\leq$ is a well founded partial order on E (the causality relation) and $\#$ is a symmetric and irreflexive relation on events (the conflict

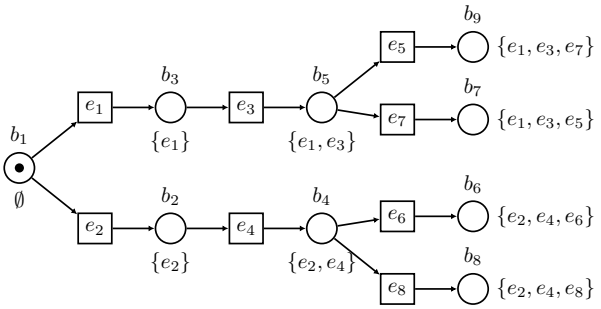


Figure 3: an occurrence net annotated on the information domain formed by subsets of events

relation) such that $e \# e' \leq e''$ implies $e \# e''$, *i.e.* the conflict relation is inherited along the causality relation. A *configuration* of a prime event structure is a subset $X \subseteq E$ of events which is *conflict free*, *i.e.* $\forall e, e' \in X.\ \neg(e \# e')$, and such that $\forall e \in X.\ \lfloor e \rfloor \subseteq X$, where $\lfloor e \rfloor$ is, as before, the subset of events $\{e' \in E \mid e' \leq e\}$. $\lfloor e \rfloor$ is a configuration itself and it is the *local* configuration. Now, given an occurrence net $C = \langle B, E, W, c_0 \rangle$, the triple $(E, \leq_C, \#_C)$ is a prime event structure and the information mapping of the spread net of Proposition 4.10 associates to each condition the local configuration of the unique event producing this condition, if such an event exists, otherwise it is the empty set.

The clear relation between prime event structures and occurrence nets is exploited on spread nets as follows: for each event e in the event structure, $\lfloor e \rfloor$ and $\lfloor e \rfloor \setminus \{e\}$ are among the elements of the support, and the operations op^i are defined for those (multi)sets of elements such that their union gives an element, namely either a local configuration or a local configuration $\lfloor e \rfloor$ without the event e. Observe that this implies that the operations are not defined for the (multi)sets of elements such that their union would contain conflicting events. The updating mapping just add an event e to a subset of events provided that this is not contained in the subset itself.

Unravel nets. Unravel nets [9] are nets in which each computation gives rise to an acyclic subnet where each place has at most one incoming arc (hence an occurrence net). Given a net $N = \langle P, T, F, m_0 \rangle$ be a net and let $T' \subseteq T$ be a subset of transitions, then $N_{T'} = \langle P', T', F', m_0' \rangle$ where $P' = P \cap ({}^\bullet T' \cup T'^\bullet) \cup m_0$, $F' = F \cap ((P' \times T') \cup (T' \times P'))$ and m_0' is m_0.

Definition 4.12. *A safe Petri net $U = \langle P, T, F, m_0 \rangle$ is an* unravel *net if it is an 1-occurrence net and for each state $X \in \mathsf{St}(N)$ the subnet $U_X = \langle {}^\bullet X \cup X^\bullet \cup m_0, X, F \cap (({}^\bullet X \cup X^\bullet \times X) \cup (X \times {}^\bullet X \cup X^\bullet)), m_0 \rangle$ is a an acyclic net where for each $p \in {}^\bullet X \cup X^\bullet$. $|{}^\bullet p| \leq 1$ and $|p^\bullet| \leq 1$.*

The requirement that the restriction of the net to the transitions in a state of the net is an acyclic net where each place has at most one incoming and one outgoing arc implies that the places in the initial marking do not have any incoming arc (as otherwise there would exists a state such that the associated net is cyclic). As an unravel net is required to be an 1-occurrence net as well, we have that each transition can be fired at some marking, and then belong at least to a state.

An unravel net as a whole may be cyclic, hence it is not possible to extract a partial order out of it but, as seen above, any run (state) gives a partial order. Places may have more than one incoming arc as well as more than one outgoing arc, thus conflicts among transitions still exist. A syntactic conflict relation is not definable, but a semantic one is available: two transitions are in conflict whenever there is no state of the net containing both. Observe that a place can be *produced* in several alternative ways.

Viewing an unravel net as a spread net is more complex than for occurrence nets or 1-occurrence nets. We can however use the fact that to each unravel net it is possible to associate a suitable event structure, the so called *bundle event structure* [25], along the observation we made on the information domain associated to an occurrence net. A bundle event structure is a triple $\mathcal{B} = (E, \vdash, \#)$ where $\# \subseteq E \times E$ is an irreflexive and symmetric relation and $\vdash \subseteq CF(E) \times E$ where $CF(E) = \{X \in \mathbf{2}^E_{fin} \setminus \{\emptyset\} \mid \forall e, e' \in X.\ \neg(e \# e')\}$. A

configuration $X \subseteq E$ is a subset of events such that X is conflict free and there exists a linearization of $e_0e_1e_2\ldots$ of the events in X such that for each i and for each $A_i \vdash e_i$ it holds that $\{e_0,\ldots,e_{i-1}\} \cap A_i \neq \emptyset$. The set of the configurations of the bundle event structure $\mathcal{B}$ is denoted with $\mathsf{Conf}(\mathcal{B})$. Given an unravel net $U = \langle P, T, F, m_0\rangle$, the triple $\mathcal{B}_U = (T, \{{}^\bullet t \vdash t \mid t \in T\}, \#)$, where $t \# t'$ iff $\forall X \in \mathsf{St}(U)\ \{t, t'\} \not\subseteq X$, is a bundle event structure. Furthermore the configurations of $\mathcal{B}_U$ are related to the states of U, namely $X \in \mathsf{St}(U)$ iff $X \in \mathsf{Conf}(\mathcal{B}_U)$. Thus, similarly to what we have observed for occurrence nets, we can use the bundle event structure to define the annotation of the places of an unravel net.

Given a bundle event structure $\mathcal{B} = (E, \vdash, \#)$, we can still associate a kind of causality relation to it. We stipulate that $e' \prec e$ whenever there exists an $A \vdash e$ such that $e' \in X$. It is then clear that, given a configuration X of this bundle event structure, $\prec^* \cap (X \times X)$ is a well founded partial order. Furthermore if $e' \prec e$ but $e' \notin X$ then there exists an $e'' \in X$ such that $e' \# e''$, $e'' \prec e$ and $e'' \in X$. With the aid of this observation we can define the set of local histories of an event. $\llbracket e \rrbracket$ is the set of subsets of events $\{X \mid e \in X,\ X \subseteq \{e' \mid e' \prec e\} \wedge X \in \mathsf{Conf}(\mathcal{B})\}$, and it is the set of local histories of the event e. With $\llbracket e \rrbracket_e$ we denote $\{X \mid X \cup \{e\} \in \llbracket e \rrbracket\}$, *i.e.* each subset in $\llbracket e \rrbracket$ without e.

Consider the unravel net $U = \langle P, T, F, m_0\rangle$, and the associated $\mathcal{B}_U = (T, \vdash_U, \#_U)$ then $\mathcal{A}$, the support of the information domain, is composed by the subsets of finite configurations of the associated bundle event structure and the $\mathsf{0}$ is $\{\emptyset\}$. The operation $\mathsf{op}^i \in \mathsf{Op}$ takes all multisets $X \in \mu\mathcal{A}$ where $X(A) > 0$ whenever there exists a place $p \in P$ such that for each $t \in {}^\bullet p$ we have that $A \neq \emptyset$ and $A \subseteq \llbracket t \rrbracket_t$, and it returns $\bigcup_{t' \in {}^\bullet p} \llbracket t' \rrbracket_{t'}$. Thus, given a transition t, the operation op^i takes all the possible local histories to which t can be added, and not only returns the local histories where t is added but also the local histories of the transitions t' (those local histories not containing t') where t' is in direct conflict with t as both put a token in the same place p. Finally $\tau(A, t)$ takes each $Y \in A$ and it either add the transition t to Y if $Y \cup \{t\} \in \mathsf{Conf}(\mathcal{B}_U)$ or it add a t' such that there exists a place $p \in P$ with $\{t, t'\} \subseteq {}^\bullet p$ and $Y \cup \{t'\} \in \mathsf{Conf}(\mathcal{B}_U)$. We denote this information domain with $\mathcal{D}_{\mathcal{B}_U}$.

We are now ready to show that also an unravel net can be considered a spread net over this information domain.

Proposition 4.13. *Let $U = \langle P, T, F, m_0, \ell\rangle$ be a label unravel net where ℓ is the identity. Let $\mathcal{B}_U = (T, \vdash_U, \#_U)$ the associated bundle event structure. Let $\mathcal{D}_{\mathcal{B}_U} = (\mathcal{A}, \mathsf{Op}, \tau, E)$ be the information domain illustrated above, and let the information mapping $h\colon P \to \mathcal{A}$ defined as $h(p) = \{X \subseteq T \mid X \in \llbracket t \rrbracket \text{ for } t \in {}^\bullet p\}$. Then $U_{\mathcal{D}_{\mathcal{B}_U}, h}$ is a spread net over $\mathcal{D}_{\mathcal{B}_U}$.*

Proof. $\mathcal{D}$ is clearly an information domain. Each place in the initial marking is annotated with the set containing just the empty set which is the $\mathsf{0}$ of $\mathcal{A}$ as, being U an unravel net, the places in the initial marking have no incoming arcs. Now consider a transition $t \in {}^\bullet p$. $\mathsf{op}^i(h({}^\bullet t))$ is defined and it is equal to the set containing the subsets of all the local histories concerning the transitions $t' \in {}^\bullet({}^\bullet t)$. Now consider a place $p \in t^\bullet$, clearly by definition of τ, we have that for all $t' \in {}^\bullet p$, $\tau(A, t) = \tau(A', t')$ for some $A, A' \in \mathsf{op}^i(h({}^\bullet t))$. Thus $U_{\mathcal{D}_{\mathcal{B}_U}, h}$ is a spread net over $\mathcal{D}_{\mathcal{B}_U}$. □

Example 4.14. *Consider the unravel net U in Figure 4. The elements of the information domain are found taking the associated bundle event structure, with events $\{e_1, e_2, e_3, e_4, e_5, e_6\}$, the conflicting events are $e_1 \#_U e_2$, $e_2 \#_U e_4$ and $e_5 \#_U e_6$, which has the following* bundles*: $\{e_1, e_2\} \vdash e_3$, $\{e_3\} \vdash e_5$, $\{e_4\} \vdash e_5$ and $\{e_3\} \vdash e_6$. The operations op^1 and op^2 are defined as follows: $\mathsf{op}^1(\{\emptyset\})$ is equal to $\{\emptyset\}$, $\mathsf{op}^1(\{\{e_1\}, \{e_2\}\})$ is equal to $\{\{e_1\}, \{e_2\}\}$ and $\mathsf{op}^1(\{\{e_1, e_3\}, \{e_2, e_3\}\})$ is equal to $\{\{e_1, e_3\}, \{e_2, e_3\}\}$, whereas $\mathsf{op}^2(\{\emptyset\}, \{\emptyset\})$ is equal to $\{\{\emptyset\}, \{\emptyset\}\}$ and finally $\mathsf{op}^2(\{\{e_4\}\}, \{\{e_1, e_3\}, \{e_2, e_3\}\})$ is equal to $\{\{e_1, e_3, e_4\}\}$. For all the other inputs they are undefined. Finally the annotations of each place p is $\bigcup_{t \in {}^\bullet p} \llbracket t \rrbracket$.*

Remark 4.15. *It should be stressed that annotating an unravel net is tricky as the annotation related to a place should contain all the local histories of the transitions in the preset, and then the operations op^i and τ are abstractly defined on the whole net. On the contrary the annotation and the operations for the occurrence and the 1-occurrence nets are much less tricky and they have a local nature rather than a global one.*

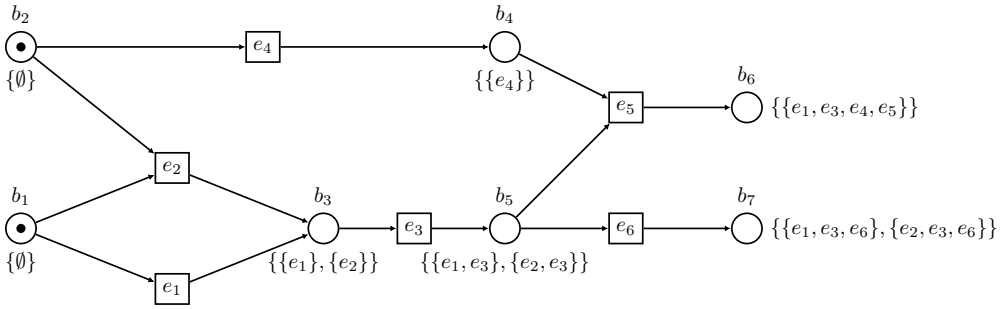


Figure 4: An unravel net

4.2. Spread nets and domains

We discuss briefly how information domains and spread nets are intertwined. Proposition 4.3 shows that each labeled safe net can be seen as a spread net over the information domain $\mathcal{D}_\perp$, namely the domain where the same information is conveyed in each place, and also the 1-occurrence net is a spread net over the same domain (Proposition 4.7). This domain is trivially regular but it is not revealing. Thus 1-occurrence nets are uniformly but not properly spread.

More appealing and challenging is the cases of occurrence nets. The relationship between the *structure* of the net and prime event structures can be exploited either to define suitable information domains or to enforce suitable structures on the net itself. We have seen how to construct an information domain tailored to an occurrence net, now we show the vice versa, namely how to construct an occurrence net starting from a suitable domain.

Proposition 4.16. *Let $(E, \leq, \#)$ be a prime event structure, let $\mathcal{D} = (\mathcal{A}, \mathsf{Op}, \tau, E)$ be the domain where $\mathcal{A} = \{X \subseteq E \mid X = \lfloor e \rfloor \setminus \{e\}\} \cup \{X \subseteq E \mid X = \lfloor e \rfloor\}$, each $\mathsf{op}^i \in \mathsf{Op}$ is defined as $\mathsf{op}^i(X) = \bigcup\{A \mid X(A) \neq 0\}$ and $\tau(A, e) = A \cup \{e\}$, and let $C_{\mathcal{D},h}$ be a spread net where C is the tuple $\langle B, E, W, c_0, \ell\rangle$ with $B = \{(\lfloor e \rfloor, e) \mid e \in E\} \cup \{(\emptyset, e) \mid e \in E\}$ and $h((X, e)) = X$, $W = \{(e, b) \mid h(b) = \lfloor e \rfloor\} \cup \{(b, e) \mid h(b) \subset \lfloor e \rfloor \wedge \forall e' \in \lfloor e \rfloor \setminus h(b).\ h(b) \not\subseteq \lfloor e' \rfloor\}$, $c_0 = \{b \in B \mid h(b) = \emptyset\}$, and ℓ is the identity. Then C is an occurrence net.*

Proof. Consider $C = \langle B, E, W, c_0, \ell\rangle$ and the information mapping $h : B \to \mathcal{A}$. As $\emptyset \in \mathcal{A}$ and $C_{\mathcal{D},h}$ is a spread net we have that all the conditions b such that $h(b) = \emptyset$ are such that ${}^\bullet b = \emptyset$. For each $b \in B$ we have that $|{}^\bullet b| \leq 1$. Assume it not the case, this means that there are at least two events e, e' in ${}^\bullet b$ and $(e, b), (e', b) \in W$, but then $\lfloor e \rfloor = h(b) = \lfloor e' \rfloor$ and as $(E, \leq, \#)$ is a prime event structure it follows that $e = e'$. The fact that W^* is a partial order follows from the fact that $\subseteq$ is a partial order, and this implies also that for each $e \in E$, the set $\{e' \mid e'\ W^*\ e\}$ is finite. Finally consider e and e which are in conflict. As the configurations of a prime event structures are such that their intersection is a configuration, also $\lfloor e \rfloor \cap \lfloor e' \rfloor$ is a configuration. Take e'' such that (a) $\lfloor e'' \rfloor \subseteq \lfloor e \rfloor \cap \lfloor e' \rfloor$ and (b) for all $\hat{e}$ that are greater than e'' we have that $\lfloor \hat{e} \rfloor \not\subseteq \lfloor e \rfloor \cap \lfloor e' \rfloor$, then we have that ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$, with $\lfloor e'' \rfloor \subset \lfloor e \rfloor$ and $\lfloor e'' \rfloor \subset \lfloor e' \rfloor$. The fact that the conflict relation of the net is inherited along the causality relation is straightforward. □

We proceed along the same line also when considering bundle event structure. Given a bundle event structure $\mathcal{B} = (E, \vdash, \#)$ and a subset $X \subseteq E$ of events, with $\mathsf{CF}(X)$ we stipulate that either $|X| = 1$ or $\forall e, e' \in X.\ e \# e'$.

Proposition 4.17. *Let $\mathcal{B} = (E, \vdash, \#)$ be a bundle event structure, and let $\mathcal{D}_\mathcal{B}$ be the domain associated to $\mathcal{B}$. Let $U_{\mathcal{D}_\mathcal{B},h}$ be a spread net where U is the tuple $\langle P, E, F, m_0, \ell\rangle$ with $P = \{(\{\emptyset\}, X) \mid X \subseteq E \wedge \mathsf{CF}(X)\} \cup \{(\mathcal{X}, X) \mid CF(X) \wedge \mathcal{X} = \bigcup_{e \in X} \lfloor\!\lfloor e \rfloor\!\rfloor\}$ and $h((\mathcal{X}, X)) = \mathcal{X}$, $F = \{(e, p) \mid \exists X \in h(p).\ e \in X\} \cup \{(p, e) \mid \exists X \in h(p).\ \exists Y \in \mathsf{Conf}(\mathcal{B}).\ X \cup \{e\} \subseteq Y \wedge \forall Z \subseteq Y \setminus (X \cup \{e\}).\ X \cup \{e\} \cup Z \in \mathsf{Conf}(\mathcal{B}) \Rightarrow X \cup \{e\} \cup Z = Y\}$, $m_0 = \{p \in P \mid h(p) = \{\emptyset\}\}$, and ℓ being the identity. Then U is an unravel net.*

Proof. To prove that U is an unravel net we have to show that each execution of the net gives an acyclic net where each place has at most one incoming arc and one outgoing arc. The places in the initial marking do not have any incoming arc by construction. Consider an event $e \in E$ and one of the local histories for it, namely $X \in \lfloor\!\lfloor e \rfloor\!\rfloor$. X can be totally ordered, *i.e.* $X = \{e_1, \ldots, e_n\}$ with $e_n = e$, with the order induced by the $\vdash$ relation. The subnet identified by the places where the information associated to them contains the proper $X_i = \{e_1, \ldots, e_i\} \subseteq X$ is a clearly acyclic and the preset and postset of each of these places contains at most one transition. This suffices to ensure that U is an unravel net. □

Proposition 4.16 and Proposition 4.17, together with Proposition 4.10 and Proposition 4.13 show that each net with certain characteristic can be turned into a spread net on a suitable domain, and also that a suitable domain induces a spread net with certain characteristic.

5. Multi-clock nets and ticking domains

In the previous section we have discussed the notion of spread net and we have shown that some of the usual nets that are used to describe a true concurrent semantics of some net N can be seen as special cases of a spread net. In this section we recall the notion of multi-clock net (see [16]) and devise a notion of information domain tailored for multi-clock nets, that we will call *ticking* domain.

5.1. Multi-clock nets

The intuition behind the notion of *multi-clock* net is the following: a net is *composed* by a number of sequential *components*, thus the *execution time* in each component of a transition of the net can be somehow inferred. Each component has a *local* clock and, as a net has various components, the net has a *multi-clock*.

A safe net $N = \langle P, T, F, m_0 \rangle$ is said to be a *net automaton* whenever each $m \in \mathcal{M}_N$ is a singleton, and each transition $t \in T$ is such that $|{}^\bullet t| = |t^\bullet| = 1$. In a net automaton only the choices between the execution of two transitions may be represented, as each reachable marking is obviously a singleton.

A multi-clock net is a safe net which can be seen as a product of net automata which *synchronize* on common transitions, and the effect of a synchronization are observable in all net automata participating to it (see transition b in Figure 5). The synchronization transitions have a preset which is not a singleton (the components to be syncronized), and we require that the cardinality of the preset of each transition is the same as the one of the postset (the effect of a synchronization, observable or not, has to take place in each of the involved components). Formally:

Definition 5.1. *A* multi-clock *net (mcn-net)* N *is the pair* (N, ν) *where* $N = \langle P, T, F, m_0 \rangle$ *is a safe net and* $\nu : P \to m_0$ *is a total mapping such that*

- *for all* $p, p' \in m_0$*, it holds that* $p \neq p'$ *implies* $\nu^{-1}(p) \cap \nu^{-1}(p') = \emptyset$*,*
- ν *is the identity when restricted to* m_0*, and*
- *for all* $t \in T$*.* ν *is injective on* ${}^\bullet t$ *and on* $t^\bullet$*, and* $\nu({}^\bullet t) = \nu(t^\bullet)$*.*

The cardinality *of a mcn-net* N*, denoted with* $\upsilon(\mathsf{N})$*, is the cardinality of* m_0*, and represents the number of components.*

As $\nu(m_0) = m_0$, ν is injective on the preset (postset) of each transition and that $\nu({}^\bullet t) = \nu(t^\bullet)$, for each $p \in m_0$, the net $N_p = \langle P_p, T_p, F_p, m_0^p \rangle$, where $P_p = \nu^{-1}(\nu(p))$, $T_p = T \cap ({}^\bullet P_p \cup P_p{}^\bullet)$, F_p is the restriction of F to P_p and T_p and $m_0^p : \overline{p} \to \mathbb{N}$ is such that $m_0^p(p') = 1$ iff $p' = p$, is a net automaton. We sometimes denote with $\overline{p}$ the subset of places P_p.

Example 5.2. *We illustrate the notion with a simple multi-clock net with just two components, depicted in Figure 5. On the left there is a multi-clock net* N *where the two components, depicted on the right,* N_{p_1} *and* N_{p_2} *synchronize on transition* b*. The two components are clearly net automata, and the* ν *mapping is* $\nu(p_1) = \nu(p_3) = p_1$ *and* $\nu(p_2) = \nu(p_4) = p_2$*.*

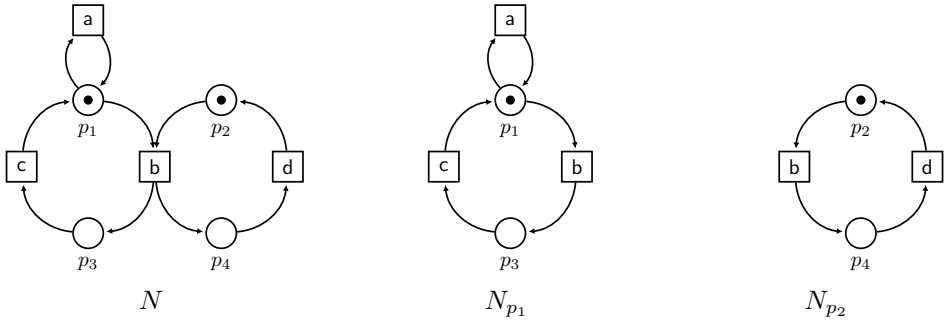


Figure 5: A multi clock net and its two components

The notion of morphism introduced for safe nets can be tailored to multi-clock nets by requiring that partitions are preserved. $(f_P, f_T) : (N, \nu) \to (N', \nu')$ is a multi-clock net morphism if (f_P, f_T) is a net morphism from N to N' and for each $p \in P$ and for each $p' \in P'$, if $p\ f_P\ p'$ then $\nu(p)\ f_P\ \nu'(p')$.

5.2. Ticking domains

As we have already said, a multi-clock net can be seen as the composition of net-automata *synchronizing* on common actions (transitions), and each component of the multi-clock net is identified via an *index* which is the unique initially marked place of the component. In other words a multi-clock net can be seen as the product of its (sequential) components. In this perspective the annotations of a spread net can be seen as the *knowledge* that each component has on its behaviour and possibly on the behaviour of the other components which may synchronize with it.

The behaviour of each single component can be easily represented as a language, as a component is a net automaton. We define the information domain associated to each component of the multi-clock net as words over transitions (labeled) of that component.

We first formalize the annotation for a single component (A is related to the (labeled) transitions of that component).

Definition 5.3. *Let* A *be an alphabet. A* ticking language *over* A *is a language* $\mathcal{L} \subseteq \mathsf{A}^*$*. The set of ticking languages over* A *will be denoted* $\mathbb{L}^{\mathsf{A}}$*.*

Given a ticking language $\mathcal{L}$ over A, $\mathsf{alph}(\mathcal{L})$ denotes its alphabet A. When clear from the context, we will omit the superscript A in $\mathbb{L}^{\mathsf{A}}$. $\mathcal{L}_0$ denotes the language reduced to the empty word ε: $\mathcal{L}_0 = \{\varepsilon\}$. A ticking language will often be obtained as the equivalence class of an equivalence relation defined on the words in A^*, and often we will use just a word to denote each element of the class.

Example 5.4. *Consider the alphabet* $\mathsf{A} = \{\mathsf{a}, \mathsf{b}\}$*, a language can be the set of all words of length* 2*, hence* $\mathcal{L} = \{\mathsf{aa}, \mathsf{ab}, \mathsf{ba}, \mathsf{bb}\}$ *or the set of all words of length less than or equal to* 3 *where all the words are equivalent to* aab *via the equivalence relation induced by the equation* $\mathsf{a} = \mathsf{b}$*, hence* $\{\mathsf{aaa}, \mathsf{aab}, \mathsf{abb}, \mathsf{bbb}, \mathsf{bab}, \mathsf{baa}, \mathsf{bba}, \mathsf{aba}\}$.

Once we have introduced the notion of ticking language, we have to say when a tuple of ticking languages can be seen as the annotation of the places of a multi-clock net. We assume the existence of a *property* P defined on tuples of languages over an alphabet A, hence $\mathsf{P} \subseteq \mathsf{A}^* \times \cdots \times \mathsf{A}^*$. A property P may be the one stating that all the words of each language have a common prefix, or that a language has all the words with the same length and each of the others languages contains just the empty word. The property will be used to sort out the tuples of languages we are interested in.

We introduce the notion of information domain tailored for multi-clock nets which we call *ticking* domain. The notion has to take into account the existence of the several components of a multi-clock net. We assume the existence of a finite set of indexes $I = \{1, \ldots, n\}$.

Definition 5.5. *Let $I = \{1, \ldots, n\}$ be a finite set of indices and let, for each $i \in I$, $\mathbb{L}_i^{\mathsf{A}_i}$ be a set of ticking languages over the alphabet A_i. Let A be $\bigcup_{i\in I} \mathsf{A}_i$ and let P be a property on the tuples with n components belonging to $(\mathsf{A}^*)^n$. Let $\mathbb{L}^{\mathsf{A}} = \bigcup_{i\in I} \mathbb{L}_i^{\mathsf{A}_i}$. A* ticking *domain $\mathcal{D}$ of arity $|I|$ over A is the quadruple $(\mathcal{A}, \mathsf{Op}, \tau, \mathsf{A})$, where*

- $\mathcal{A} = \{v : I \to \mathbb{L}^{\mathsf{A}} \mid v(i) \in \mathbb{L}_i^{\mathsf{A}_i} \wedge \mathsf{P}(v(1), \ldots, v(n))\}$*, the elements of $\mathcal{A}$ are called* vector-clock *and are ranged over by v, and $\mathsf{0}$ is the vector-clock v such that $v(i) = \mathcal{L}_0$ for each $i \in I$,*
- Op *is a set of partial operations $\mathsf{op}^i : \mathcal{A}^i \rightharpoonup \mathbf{2}^{\mathcal{A}}$ such that*
  - *$i \leq |I|$, and*
  - *if $\mathsf{op}^i(v_1, \ldots, v_i)$ is defined then $\mathsf{op}^i(v_1, \ldots, v_i) \neq \emptyset$, and*
- *$\tau : I \times \mathcal{A} \times \mathsf{A} \to \mathcal{A}$ is the* ticking *mapping that takes an index $i \in I$, a vector clock and an element in A and returns a vector clock.*

The notion of ticking domain is an obvious adaptation of Definition 3.2 to the annotation of multi-clock nets. The elements of a ticking domain are vectors of languages (or words) where the components of the vector all together satisfy a property P, we will see its use in the next sections. Each operation $\mathsf{op}^i \in \mathsf{Op}$ acts on a multiset of vector-clocks containing i elements, that cannot be more than the cardinality of the ticking domain. The unique relevant difference from Definition 3.2 is the updating mapping, which is called here ticking mapping. It acts on each component separately, and the effects may differ from a component to another. We again say that a ticking domain is regular is the operations in Op return a singleton. In this case we write $\mathsf{op}^i : \mathcal{A}^i \rightharpoonup \mathcal{A}$.

As before, given the set of vector-clocks, the set of operations and the ticking mapping (*i.e.* the updating mapping) will be defined when constructing the proper domains for the spread net we are interested in. Given a ticking domain $\mathcal{D}$ of arity $|I|$, with $\mathsf{A}_{i,j}$ we denote the alphabet $\mathsf{A}_i \cap \mathsf{A}_j$, with $i, j \in I$ (obviously $\mathsf{A}_{i,j} = \mathsf{A}_{j,i}$). We will sometime omit the word ticking when it is clear from the context that we are referring to a ticking domain.

We already said that the languages of a vector-clock v may be induced by a suitable equivalence relation $\sim$. In the case that the equivalence relation is well understood, rather than writing $v(i) = \{w' \mid w' \sim w\} = \mathcal{L}_{\sim}^{w}$ we simply write $v(i) = w$ and also $(w_1, \ldots, w_n)$ to denote the vector-clock where $v(i) = \mathcal{L}_{\sim}^{w_i}$ for $1 \leq i \leq n$.

Example 5.6. *We first set a generic domain. Take any finite index set I and consider the set of alphabets A_i, with $i \in I$, then a ticking-domain could be $(\mathcal{A}, \mathsf{Op}, \tau, \bigcup_{i\in I} \mathsf{A}_i)$ with $\mathcal{A}$ being the set of vectors v where each component $v(j)$ is the language with just one word and the property P could be that $\forall j, k \in I$, $\forall w \in v(j)$, $\forall u \in v(k)$, $proj(w, \mathsf{A}_{j,k}) = proj(u, \mathsf{A}_{j,k})$ (the elements of the vector-clock* agree *on the common symbols). The operations can be defined as follows. Take a tuple $(v_1, \ldots, v_i)$ of vector-clocks. The operations are defined only if for each pair of vector-clock v_n and v_k, the elements are such that for each $m \in I$ either the word of the language $v_n(m)$ is a prefix of the word of the language $v_k(m)$ or vice versa. The result is obtained by taking the* maximum *of all these words. The operation devised in this way is well defined, as it returns a vector-clock satisfying P. Finally the ticking mapping, for each index, just concatenate the to each word in each language the element of the alphabet $\bigcup_{i\in I} \mathsf{A}_i$, provided that it belongs to the alphabet of the language.*

Concretely consider two alphabets $\mathsf{A}_2 = \{\mathsf{a}, \mathsf{b}\}$ and $\mathsf{A}_1 = \{\mathsf{b}, \mathsf{c}\}$, the elements of the domain are the pairs of words (w, u) having the same number of b (we identify the language with the unique word it contains), and the operations either take a vector-clock and it returns the vector-clock itself or it take two vector-clocks $v = (w, u)$ and $v' = (w', u')$. In these two vector-clocks it should be that either w is a prefix of w' or vice versa, and that either u is a prefix of u' or vice versa, and the operation returns a vector-clock composed by the two longest words. We show that this operation is well defined and the resulting vector-clock satisfies the property. Assume that w is a prefix of w' and that u' is a prefix of u, and assume that $proj(u, \{\mathsf{b}\}) \neq proj(w', \{\mathsf{b}\})$. Then, as $\mathsf{P}(v')$ holds, we have that $proj(w', \{\mathsf{b}\}) = proj(u', \{\mathsf{b}\})$, thus necessarily w contains fewer b than w', but also $proj(w, \{\mathsf{b}\}) = proj(u, \{\mathsf{b}\})$ as $\mathsf{P}(v)$ holds, hence it cannot

be that u' is a prefix of u as $proj(u, \{\mathsf{b}\}) \geq proj(u', \{\mathsf{b}\})$. It holds then that $proj(u, \{\mathsf{b}\}) = proj(w', \{\mathsf{b}\})$ and $\mathsf{P}((w', u))$ holds. Finally τ, for each index, just concatenate the element to the words if the element belongs to the alphabet.

We stress the fact that, as multi-clock nets can be seen as the composition of more elementary components, also a ticking domain can be seen as the composition of simpler components. The main difference is that we have to establish when the pieces of information coming from other components are *coherent* and this is the purpose of the predicate P.

6. Unfoldings of Multi-clock nets as spread nets

When illustrating the notion of spread net (Section 4) we have shown that various subclasses of nets, each of them modeling a semantics for (safe) nets, could be viewed as spread nets for suitable information domains, which we have constructed having in mind how we wanted to annotate these nets. In this section we explore various notions of unfoldings in the same manner. We focus in particular on unfoldings of multi-clock nets. For various notions of unfoldings proposed in the literature, we adapt them to multi-clock nets and show how they can be seen as spread nets over suitable domains that are always revealing ones. The element of the ticking domain associated to each place (condition) of the unfolding will carry information about the knowledge that each component has on the executions leading to that place (condition), which is inferred from the way the net is unfolded.

For each kind of unfolding we characterize the associated ticking domain, and each element of a ticking domain will be a vector-clock where each entry is a language (possibly induced by an equivalence class) of words on the alphabet defined by the transitions of the corresponding component of the multi-clock net to be *unfolded*.

We recall when we consider a net an *unfolding*. We say that a net N' is the unfolding of another net N if some conditions are fulfilled:

- each transition of the unfolding is executed at most once in a computation,
- each computation of the unfolding can be mapped to a computation in the original net via a *folding* morphism, and
- to each computation of the original net it is possible to find a *corresponding* computation in the unfolding (where corresponding means that the former computation is mapped to the latter one).

Thus an unfolding is a net N' (possibly enjoying some specific characteristics like, for instance, being an unravel net or an occurrence net) and a suitable labeling mapping that turns to be a *folding* morphism.

Let $N = \langle P, T, F, m_0 \rangle$ and $N' = \langle P', T', F', m_0' \rangle$ be two safe nets, a folding morphism $f \colon N \to N'$ is a net morphism where the following further conditions hold:

1. $f \colon P \to P'$ and $f \colon T \to T'$ are total mappings,
2. $\forall t \in T$ it holds that f is bijective on ${}^\bullet t$ and on $t^\bullet$ as well, and
3. $\forall t, t' \in T$ if ${}^\bullet t = {}^\bullet t'$ and $f(t) = f(t')$ then $t = t'$.

The first condition ensures that each place (condition) and transition (event) of the unfolding have a counterpart, *i.e.* they correspond to a place or a transition of the net to be unfolded, the second states bijectivity on the presets or postsets of transitions and the last requirement is a *parsimony* requirement: the net representing an unfolding does not contain two transitions which are *indistinguishable* (bearing the same label and using the same conditions, thus producing the same conditions). In the case of multi-clock nets the morphisms (and henceforth the folding morphisms as well) preserve the partitions.

In this section with $\mathsf{alph}(N_p)$ we will denote the set $T_p \subseteq T$ of the subnet N_p of the multi-clock net $\mathsf{N} = (N, \nu)$, with $N = \langle P, T, F, m_0 \rangle$, and $p \in m_0$.

6.1. 1-unfolding of a multi-clock net

The notion of 1-unfolding has been formalized by van Glabbek and Plotkin in [26] and [24] and it is meant to capture the so called *collective* token philosophy.

We briefly illustrate the idea behind this construction. In the collective-token philosophy it is irrelevant to keep track of the transitions that have produced the tokens needed to execute a transition, all the different histories are *equated*. The only requirement is that each occurrence of a transition in a computation of the original net is uniquely represented in the 1-unfolding. It is then enough to create, for each transition, an *occurrence* of it connected to the places as the original transition and enforce that this occurrence is executed just once. In the construction this is achieved by *creating* a place in the preset of each transition and guaranteeing that this place gets marked just once. Occurrences of the same transition are *numbered* and we implement their ordered execution. The construction is rather syntactic and, to assure that each transition of the 1-unfolding can be executed, we assume that each transition of the multi-clock net N to be unfolded can be executed an arbitrary number of times. Henceforth we assume the multi-clock net $\mathsf{N} = (N, \nu)$ is *live*.

The only information that the 1-unfolding has to convey regards the ordering in the execution of a single transition. To achieve this we *enrich* the multi-clock net by adding some components that have this *numbering* purpose. Furthermore this enrichment will allow us to define a proper folding morphism.

Enriching multi-clock nets. The enrichment is done by adding a new component for each transition of the original net, and the added component has just a *numbering* purpose: it should guarantee that the executions of each transition in the net can be numbered just by looking at this added component. We will call the added components *numbering*-automata.

Consider a multi-clock net $\mathsf{N} = (N, \nu)$, where $N = \langle P, T, F, m_0 \rangle$, for each $t \in T$ we add a place p_t which is initially marked and connected with t with an incoming and an outgoing arc.

Definition 6.1. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net, where N is $\langle P, T, F, m_0 \rangle$. Then the* enrichment *of N, denoted with $\widetilde{\mathsf{N}}$, is the pair $(\widetilde{N}, \widetilde{\nu})$ where*

- *$\widetilde{N} = \langle \widetilde{P}, T, \widetilde{F}, \widetilde{m_0} \rangle$ is a safe net where*
  - *$\widetilde{P} = P \cup \{p_t \mid t \in T\}$,*
  - *$\widetilde{F} = F \cup \{(p_t, t) \mid t \in T\} \cup \{(t, p_t) \mid t \in T\}$. and*
  - *$\widetilde{m_0}(p) = m_0(p)$ when $p \in P$ and $\widetilde{m_0}(p) = 1$ when $p \in \{p_t \mid t \in T\}$, and*
- *$\widetilde{\nu}$ extends ν stipulating that $\widetilde{\nu}(p_t) = p_t$.*

If the original multi-clock net was formed by n synchronizing net automata and it has m transitions, the enriched multi-clock net is formed by $m + n$ synchronizing net automata and the added automata $N_{p_t} = (\{p_t\}, \{t\}, \{(p_t, t), (t, p_t)\}, \{p_t\})$ (we confuse the initial marking with the set) simply *counts* the occurrence of the transition t.

Proposition 6.2. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net and $\widetilde{\mathsf{N}} = (\widetilde{N}, \widetilde{\nu})$ its enrichment. Then $\widetilde{\mathsf{N}}$ is a multi-clock net.*

It is a trivial observation that there is a one to one correspondence between the reachable markings of N and those of $\widetilde{N}$, and between the firing sequences as well.

Proposition 6.3. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net and $\widetilde{\mathsf{N}} = (\widetilde{N}, \widetilde{\nu})$ its enrichment. Let $\widehat{m}$ defined as as $\widehat{m}(p) = m(p)$ if $p \in P$ and $\widehat{m}(p_t) = 1$, and let its extension to firing sequences defined as $\widehat{m\,[t\rangle\,\sigma} = \widehat{m}\,[t\rangle\,\widehat{\sigma}$. Then $\widehat{\mathsf{FS}^N_{m_0}} = \mathsf{FS}^{\widetilde{N}}_{\widetilde{m_0}}$ and $\widehat{\mathcal{M}_N} = \mathcal{M}_{\widetilde{N}}$.*

Proof. Obvious. □

The enrichment does not influence the behaviour of a multi-clock net.

1-unfolding construction. As we have done for occurrence and unravel nets, we call places conditions and transitions events, thus we use B and E for condition and events, respectively.

Definition 6.4. *Let $\mathsf{N} = (N, \nu)$, with $N = \langle P, T, F, m_0 \rangle$, be a multi-clock net and let $\widetilde{\mathsf{N}}$ be its enrichment. The 1-unfolding $\mathcal{O}$ is the (labeled) multi-clock net (O, f), with $\mathsf{O} = (O, \nu_O)$, where $O = \langle B, E, W, c_0, f_T \rangle$, ν_O and f are defined as follows:*

- $B = (P \times \{*\}) \cup (\{p_t \mid t \in T\} \times \mathbb{N})$,
- $E = T \times (\mathbb{N} \setminus \{0\})$,
- $W = \{((p,*),(t,i)) \mid p \in P, t \in T, (p,t) \in F \text{ and } i \in \mathbb{N}\} \cup \{(((t,i),p,*)) \mid p \in P, t \in T, (t,p) \in F \text{ and } i \in \mathbb{N}\} \cup \{((p_t,i),(t,i+1)) \mid t \in T \text{ and } i \in \mathbb{N}\} \cup \{((t,i),(p_t,i)) \mid t \in T \text{ and } i \in \mathbb{N}\}$,
- $c_0((p,*)) = m_0(p)$, $c_0((p_t,0)) = 1$ *and* $c_0((p_t,i)) = 0$ *for* $i \neq 0$,
- $\nu_O((p,*)) = \nu(p)$ *and* $\nu_O((p_t,i)) = (p_t,0)$, *and*
- $f_P((p,*)) = p$, $f_P((p_t,i)) = p_t$, *and* $f_T(t,i) = t$.

The 1-unfolding $\mathcal{O}$ is related to the multi-clock net $\widetilde{N}$, and then to N itself. Each condition $(p,*)$ is related to the place p and the condition (p_t,i) to the place p_t and the event (t,i) to the transition t. It is routine to check that f is a morphism (which implies that each computation of the 1-unfolding O is mapped to a computation of the multi-clock net $\widetilde{N}$ and of N as well). The requirement that each computation of $\widetilde{N}$ has a correspondence in a computation of O is trivial. We have turned the construction into a labeled net as the f_T can be seen as a labeling of event: to each event (t_i,k) it associates the *label* t.

The liveness assumption on the multi-clock net N allows us to create as many copies of t as we need without being forced to count them beforehands, as the net can stutter. Would we avoid the requirement that each transition of N can be executed an arbitrary number of times we would have to calculate, for each transition t, its maximal number of occurrences resorting to the states of the multi-clock net N. Assume, for instance, that the transition T is executed at most n times in each state, *i.e.* $n = \mathsf{max}(\{X(t) \mid X \in \mathsf{St}(N)\}) = \xi(t)$, then we have to introduce the places $\{(p_t,i) \mid i \leq \xi(t)\}$, the transitions associated to t would be $\{(t,i) \mid 1 \leq i \leq \xi(t)\}$ and so forth. The way places and transitions are connected would not change, as well as the other ingredients of the construction.

$\mathsf{O} = (O, \nu_O)$ is indeed a multi-clock net and O is as well an occurrence net, *i.e.* each event (transition) is executed just once.

Proposition 6.5. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net and $\widetilde{\mathsf{N}}$ its enrichment. Let $\mathcal{O} = ((O, \nu_O), f)$ be its 1-unfolding. Then (O, ν_O) is a multi-clock net and f is a folding morphism.*

Proof. The fact that f is a morphism trivial (it is the identity relation on the places in P and all the *numbering* places are related to the proper numbering one), and it is also a folding morphism: f is clearly total on places and transitions, and the other two conditions are trivial. Clearly each component of the 1-unfolding is a net-automata: the one deriving by the ones non numbering one are net-automata as they are net automata in N, and the numbering ones are net-automata as well. □

Given an 1-unfolding $\mathcal{O} = (\mathsf{O}, f)$, we now discuss on how to turn O into a spread net. The ticking domain tailored for this construction is the one where each vector-clock either consists of just the empty word for each entry of the vector-clock, or it has just one entry which is not the empty word and this entry is associated to a *numbering* component of the enrichment. The operations and the ticking mapping are defined according to this intuition.

Example 6.6. *The multi-clock net $\mathsf{N} = (N, \nu)$ on the left in Figure 6 has just one component with two transitions. Its enrichment $\widetilde{N}$ is shown below N and has three components, where two of them are numbering ones. On the right of these two multi-clock nets we depict the initial part of the 1-unfolding. Observe that the piece of information associated to the conditions of the unfolding carry, for each numbering component, the number of timess the associated transition has occurred.*

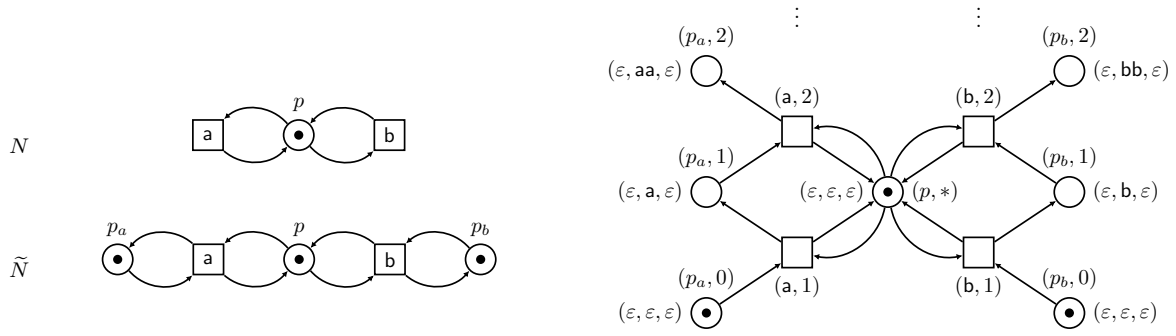


Figure 6: A multi-clock net with three components and its 1-unfolding

The following definition details the discussion on how to define the domain.

Definition 6.7. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net, where $N = \langle P, T, F, m_0 \rangle$ and $n = |T|$, and let $\widetilde{\mathsf{N}}$ its enrichment. The ticking domain for the 1-unfolding of N, denoted with $\mathcal{D}_{\mathsf{N}}^{1unf}$ is the quadruple $(\mathcal{A}, \mathsf{Op}, \tau, T)$ where*

- $\mathcal{A} = \{(w_1, \ldots, w_n, u_{p_{t_1}}, \ldots, u_{p_{t_n}}) \mid \exists!\ p_{t_k}.\ u_{p_{t_k}} \neq \varepsilon \ \wedge\ \forall i \neq p_{t_k}.\ w_i = \varepsilon\} \cup \{(\varepsilon, \ldots, \varepsilon)\}$, *with the* $\mathsf{0}$ *being the vector-clock* $(\varepsilon, \ldots, \varepsilon)$,
- $\mathsf{Op} = \{\mathsf{op}^k \mid 2 \leq k \leq n+1\}$ *where* $\mathsf{op}^k(v_1, \ldots, v_k)$ *is defined when all the* k *vector-clock are* $\mathsf{0}$ *or just one of them is not, and*

  $$\mathsf{op}^k(v_1, \ldots, v_k) = \begin{cases} \mathsf{0} & \text{if } \forall i \leq k.\ v_i = \mathsf{0} \\ v_j & \text{if } \exists!\ j. v_j \neq \mathsf{0} \end{cases}$$

  i.e. *the result is* $\mathsf{0}$ *in the case all the vector-clocks are* $\mathsf{0}$ *and the only one not being* $\mathsf{0}$ *otherwise, and*
- τ *is defined as follows:*

  $$\tau(i, (w_1, \ldots, w_{n+m}), t_k) = \begin{cases} \mathsf{0} & \text{if } i \neq p_{t_k} \\ (\varepsilon, \ldots, w_{p_{t_k}} \cdot t_k, \ldots, \varepsilon) & \text{otherwise} \end{cases}.$$

We stress that $\exists!\ p_{t_k}.\ u_{p_{t_k}} \neq \varepsilon \ \wedge\ \forall i \neq p_{t_k}.\ w_i = \varepsilon$ is the property the elements of the ticking domain have to fulfill, and it says what we have stressed before, namely that at most one component of the vector-clock is different from $\mathcal{L}_0$, and this component is one of the numbering components.

Proposition 6.8. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net, where $N = \langle P, T, F, m_0 \rangle$, and let $\widetilde{\mathsf{N}}$ its enrichment. Let $\mathcal{O} = (\mathsf{O}, f)$ be its 1-unfolding. Then $\mathsf{O}_{\mathcal{D}_{\mathsf{N}}^{1unf}, h}$ where*

- $\mathsf{O} = (O, \nu_O)$ *is the labeled multi-clock net where $O = \langle B, E, W, c_0, f_T \rangle$ is obtained by the multi-clock (O, ν_O) constructed in Definition 6.4, and*
- $h \colon B \to \mathcal{A}$ *is defined as*

  $$h(b) = \begin{cases} \mathsf{0} & \text{if} \quad b \in P \\ (\varepsilon, \ldots, t^i, \ldots, \varepsilon) & \text{if} \quad b = (p_t, i) \end{cases}$$

is a spread net over $\mathcal{D}_{\mathsf{N}}^{1unf}$.

Proof. By construction it is easy to see that the conditions of Definition 4.1 are fulfilled. □

We observe that the ε in the vector-clock corresponding to a non numbering component can be considered as the representative of the equivalence defined equating all the local computation to the empty one (thus forgetting everything). The words in the numbering components are those used to guarantee the order in the execution of the same transition (each word is equivalent to itself).

6.2. Branching processes

The 1-unfolding realizes the idea that, in each component which is not a numbering one, the whole past of each place is forgotten. This is represented by the $\mathsf{0}$ annotation of each condition beside the ones belonging to numbering components. On the contrary we can imagine that the *whole* past is kept. This gives rise to branching process ([4] and [5]). The notion is based on the one of occurrence net where, as we have already mentioned, the computations are definable without resorting to the firing sequence. We recall that in aa occurrence net $C = \langle B, E, W, c_0\rangle$ a subset of conditions $X \subseteq B$ is said to be *concurrent* whenever $\forall b, b' \in X.\ b \neq b' \ \Rightarrow\ (\neg(b \leq_C b') \ \wedge\ \neg(b' \leq_C b) \ \wedge\ \neg(b \#_C b'))$, and it is denoted with $\mathsf{co}(X)$. We define at the same time the occurrence net and the folding morphism

Definition 6.9. *Let $\mathsf{N} = (N, \nu)$, with $N = \langle P, T, F, m_0\rangle$, be a multi-clock net. The unfolding $\mathcal{C}$ is the labeled multi-clock net (C, f), with $\mathsf{C} = (C, \nu_C)$, where $C = \langle B, E, W, c_0, f_T\rangle$, ν_C and f are defined as follows:*

- $B = \{(m_0, p) \mid p \in m_0\} \cup \{(\{e\}, p) \mid e \in E \ \wedge\ p \in f_T(e)^\bullet\}$,
- $E = \{(X, t) \mid X \subseteq B \ \wedge\ \mathsf{co}(X) \ \wedge\ {}^\bullet t = f_P(X)\}$,
- $W = \{(b, e) \mid e = (X, t) \ \wedge\ b \in X\} \cup \{(e, b) \mid b = (\{e\}, p)\}$,
- $c_0(b) = 1$ *if* $b = (m_0, p)$ *and* $c_0(b) = 0$ *otherwise,*
- $\nu_C((-, p)) = \nu(p)$, *and*
- $f_P((-, p)) = p$ *and* $f_T((-, t)) = t$.

The morphism $f\colon C \to N$ is clearly a folding morphism, and the resulting net is obviously an occurrence net which is a multi-clock net as well. The unique difference with respect to the classical notion is that we add the labeling to the occurrence net itself.

Proposition 6.10. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net. Let $\mathcal{C} = ((C, \nu_C), f)$ be its unfolding. Then (C, ν_C) is a multi-clock net and f is a folding morphism.*

Proof. See [4] and [16]. □

We now discuss on how to turn the multi-clock net (C, ν_C) which is the one defined in Definition 6.9 into a spread net, using the fact that C is an occurrence net. Take the labeled occurrence multi-clock net $\mathsf{C} = (C, \nu_C)$ where $C = \langle B, E, W, c_0, \ell\rangle$ and the labels are the set T, and consider the condition $b \in B$. Assume that ${}^\bullet b \neq \emptyset$, hence it is caused by a unique event, say e, as ${}^\bullet b$ is a singleton. Consider now the set of events $X_e = \{e' \mid e' \leq_C e\}$, and let $x_e \in E^*$ any linearization of the events in X_e compatible with $\leq_C$ (meaning that if $e' \leq_C e''$ then e' precedes e'' in x_e) and $f_T(w_e) \in T^*$. We observe that, for each $p \in m_0$:

1. $w_p = proj(f_T(x_e), T_p)$ is a word over $\mathsf{alph}(N_p)$ and there exist $\sigma \in \mathsf{FS}^{N_p}_{m_0}$, such that $\mathsf{tr}(\sigma) = w_p$, *i.e.* projecting the word $f_T(x_e)$ on $\mathsf{alph}(N_p)$ we obtain a trace of the component N_p, and
2. let $e', e'' \in \{\!|x_e|\!\}$ such that $f_T(e'), f_T(e'') \in \mathsf{alph}(N_p)$, then either $e' \leq_C e''$ or $e'' \leq_C e'$, *i.e.* two events belonging to the same component are ordered.
3. let $B_e = \{b \mid b \leq_C e\}$, clearly there exists a $b_0 \in c_0$ and $b_0 \in B_e$. Consider all the conditions belonging to the same component, and call them B_p for $p \in m_0$. Then $B_e \cap B_p$ is a totally ordered set and provided that $f_P(b_0) = p \in m_0$, we have that $\{\!|proj(f_T(w_e), T_p)|\!\} = (B_e \cap B_p)^\bullet$.

It is then clear what could be the annotation associated to each condition b of this occurrence net: it is a tuple $(w_{p_1}, \ldots, w_{p_1})$ where each $w_{p_i} = proj(f_T(x_e), T_p)$ with $e \in {}^\bullet b$ and $\{p_1, \ldots, p_n\} = m_0$. The w_{p_i} does not depend on a linearization of X_e, *i.e.* for any two different linearization x and x' of X_e compatible with the partial order induced by the occurrence net, it holds that $proj(f_T(x), T_p) = proj(f_T(x'), T_p)$. The property P that each vector-clock $(w_1, \ldots, w_n)$ has to satisfy can be written as follows: $\forall i, j \in \{1, \ldots, n\}$. $proj(w_i, \mathsf{A}_j) = proj(w_j, \mathsf{A}_i)$. Thus $\mathcal{A} = \{(w_1, \ldots, w_n) \mid w_i \in \mathsf{A_i}^* \ \wedge\ \forall i, j.\ proj(w_i, \mathsf{A}_j) = proj(w_j, \mathsf{A}_i)\}$

Consider now a set of vector-clock $\{v_1, \ldots, v_k\}$. We say that $\{v_1, \ldots, v_k\}$ is *bp-homogeneous* if for all i, k and for all j it holds that either $v_i(j)$ is a prefix of $v_k(j)$ or $v_k(j)$ is a prefix of $v_i(j)$. In this case it is possible to define a maximum among these words, which we denote by $\mathsf{max}(\{v_1(j), \ldots, v_k(j)\})$. The operation $\mathsf{op}^i(\{v_1, \ldots, v_i\})$ is defined for bp-homogeneous tuples of vector-clocks and it returns the vector-clock $(\mathsf{max}(\{v_1(i), \ldots, v_i(1)\}), \ldots, \mathsf{max}(\{v_1(n), \ldots, v_i(n)\}))$. The result satisfies the property P. Assume it does not, then there exist two indexes of the resulting vector-clock, say k and j, such that such that $proj(w_k, \mathsf{A}_j) = proj(w_j, \mathsf{A}_k)$. Now w_k is the $\mathsf{max}(\{v_1(k), \ldots, v_i(k)\})$ and w_j is the $\mathsf{max}(\{v_1(j), \ldots, v_i(j)\})$, and, arguing like we did in Example 5.6, we arrive at a contradiction. Hence the operations are well defined. Finally $\tau(i, v, t)$ just concatenates the symbol t to each word on an alphabet containing t.

Definition 6.11. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net, where $N = \langle P, T, F, m_0 \rangle$. The ticking domain for the branching process of N is the quadruple $(\mathcal{A}, \mathsf{Op}, \tau, T)$ where*

- $\mathcal{A} = \{(w_1, \ldots, w_n) \mid w_i \in \mathsf{A_i}^* \ \wedge \ \forall i, j.\ proj(w_i, \mathsf{A}_j) = proj(w_j, \mathsf{A}_i)\}$ *with* $\mathsf{0} = (\varepsilon, \ldots, \varepsilon)$,
- $\mathsf{Op} = \{\mathsf{op}^i \colon \mathcal{A}^i \to \mathcal{A} \quad | \quad 1 \leq i \leq v(\mathsf{N})\}$ *where* $\mathsf{op}^k(v_1, \ldots, v_k)$ *is defined when* $\{v_1, \ldots, v_i\}$ *is bb-homogeneous and* $\mathsf{op}^k(v_1, \ldots, v_k) = (\mathsf{max}(V_1), \ldots, \mathsf{max}(V_n))$ *where* V_k *is the set* $\{v_i(k) \mid v_i \in V\}$, *and*
- $\tau : I \times \mathcal{A} \times T \to \mathcal{A}$, *where* $I = m_0$, *takes and index, a vector-clock* v *and an element* $t \in T$ *and returns the vector-clock* $(w_1, \ldots, w_n)$ *where* $w_i = v(i) \cdot t$ *if* $t \in T_i$ *and* $w_i = v(i)$ *otherwise,*

and it is denoted with $\mathcal{D}^{bp}_{\mathsf{N}}$,

We are now ready to show that the unfolding of a multi-clock net is indeed a spread net.

Proposition 6.12. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net, where $N = \langle P, T, F, m_0 \rangle$. Let $\mathcal{C} = (\mathsf{C}, f)$ be its unfolding, where $\mathsf{C} = (C, \nu_C)$. Then $\mathsf{C}_{\mathcal{D}^{bp}_{\mathsf{N}}, h}$ where*

- $\mathsf{C} = (C, \nu_C)$ *is the labeled multi-clock occurrence net where $C = \langle B, E, W, c_0, f_T \rangle$ and ν_C are as in Definition 6.9, and*
- $h \colon B \to \mathcal{A}$ *is defined as $h(b) = (w_{p_1}, \ldots, w_{p_p})$ where $w_{p_1} = proj(f_T(x_e), T_{p_1})$ and X_e is any linearization of $\{e' \mid e' \leq_C e\}$ compatible with $\leq_C$},*

is a spread net over $\mathcal{D}^{bp}_{\mathsf{N}}$.

Proof. Again by construction it is easy to see that all the conditions of Definition 4.1 are fulfilled. In particular take a subset $X \subseteq B$ such that $\mathsf{co}(X)$ and there exists an event e such that ${}^\bullet e = X$, then it is easy to observe that the set $\{h(b) \mid b \in X\}$ is a bp-homogeneous set of vector-clocks, and this guarantees that the vector-clocks associated to the conditions in the postset of e are the correct ones. □

Example 6.13. *Consider the multi-clock net in the Example 5.2. In Figure 7 we show the initial part of its branching process, and the annotations making it a spread net.*

6.3. Trellis processes

Looking from the point of view of how much information on executions is preserved, the *1-unfolding* conveys the idea that the whole past is forgotten, hence also dependencies and conflicts are identified (the only information conveyed is that the i-th occurrence of a transition is preceded by the $i-1$-th occurrence of the same transition). Somehow we can say that *1-unfoldings* are *minimally spread*. Instead the notion of *branching process* captures the intuition that all the dependencies and conflicts are properly presented. We can say that *branching process* are *maximally spread*.

In a multi-clock net two transitions belonging to the same automaton are never concurrent, they may be in conflict in the case they have a common place in their preset or postset. These conflicts are somehow local

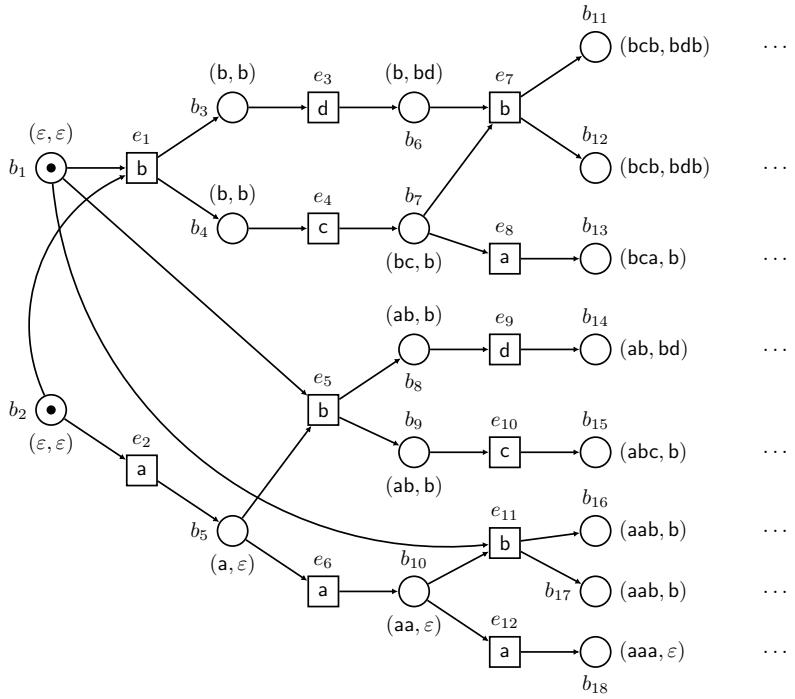


Figure 7: The spread net obtained by the initial part of the branching process of Figure 5

to the component involved. The idea of a *trellis process* is that local conflicts may be identified provided that the conflicting transitions happen at the same time and determine the same local state, *i.e.* the conflicting transitions mark the same place. Thus trellis processes equate some of the computations of the underlying multi-clock net. Here we give a direct construction in the style of *1-unfolding* of *branching processes* seen before.

We introduce a number of auxiliary notions. Given an unravel net $N = \langle P, T, F, m_0\rangle$, we say that $P' \subseteq P$ are *concurrent* places whenever there exists a firing sequence $\sigma \in \mathsf{FS}^N_{m_0}$ such that $P' \subseteq \mathit{lead}(\sigma)$. We denote a set of concurrent place P' with $\hat{\mathsf{co}}(P')$. With respect to the analogous notion in occurrence nets, this is semantic notion and clearly if $N = \langle P, T, F, m_0\rangle$ is an occurrence net, then it holds that $\mathsf{co}(X)$ iff $\hat{\mathsf{co}}(X)$ for any subset of places X of P. The second one will be used to construct the conditions of the trellis. Consider the multi-clock net $\mathsf{N} = (N, \nu)$, with $N = \langle P, T, F, m_0\rangle$. With $\mathfrak{T}^i_p$ we denote the set $\{w \in T^*_p \mid |w| = i \ \wedge\ \exists \sigma \in \mathsf{FS}^{N_p}_{m^p_0}.\ \mathsf{tr}(\sigma) = w\}$, *i.e.* all the traces of length i in the net automaton N_p. Finally with P^i_p we denote the set of places of N_p which are marked with a firing sequence of length i, namely $\{p' \in P_p \mid \exists \sigma \in \mathsf{FS}^{N_p}_{m^p_0}.\ |\mathsf{tr}(\sigma)| = i \ \wedge\ p' \in \mathit{lead}(\sigma)\}$. With the aids of these notion we can introduce the notion of *trellis* process

Definition 6.14. *Let* $\mathsf{N} = (N, \nu)$*, with* $N = \langle P, T, F, m_0\rangle$*, be a multi-clock net. The* trellis process $\mathfrak{U}$ *is the (labeled) multi-clock net* (U, f)*, with* $\mathsf{U} = (U, \nu_U)$*, where* $U = \langle B, E, W, c_0, f_T\rangle$*,* ν_U *and* f *are defined as follows:*

- $B = \{(p', \{\varepsilon\}) \mid p' \in m_0\} \cup \big(\bigcup_{i>0}\{(p', \mathfrak{T}^i_p) \mid p' \in P^i_p\}\big)$,
- $E = \{(X, t) \mid X \subseteq B \ \wedge\ \hat{\mathsf{co}}(X) \ \wedge\ {}^\bullet t = f_P(X)\}$,
- $W = \{(b, e) \mid e = (X, t) \ \wedge\ b \in X\} \ \cup\ \{(e, b) \mid e = (X, t),\ b = (p', \mathfrak{T}^i_p) \ \wedge\ \exists\ b' \in X.\ (b' = (p'', \mathfrak{T}^{i+1}_p) \ \wedge\ p' \in t^\bullet)\}$,

- $c_0(b) = 1$ *if* $b = (p, \{\varepsilon\})$ *and* $c_0(b) = 0$ *otherwise,*
- $\nu_C((-, p)) = \nu(p)$*, and*
- $f_P((p, -)) = p$ *and* $f_T((-, t)) = t$*.*

The morphism $f\colon U \to N$ is clearly a folding morphism, and the resulting net is an unravel net which is a multi-clock net as well.

This construction mimics the one of branching process seen previously, following the intuition given in [16]. Thus an event has in its preset a set of concurrent conditions that are mapped onto the preset of the image of the event. The conditions in the postset of the event are calculated using the set of traces associated to the conditions in the preset.

Proposition 6.15. *Let* $\mathsf{N} = (N, \nu)$ *be a multi-clock net. Let* $\mathcal{U} = ((U, \nu_U), f)$ *be its trellis process. Then* (U, ν_U) *is a multi-clock net and* f *is a folding morphism.*

Proof. The fact that (U, ν_U) where $U = \langle B, E, W, c_0, f_T\rangle$ is a multi-clock net follows at once by the construction. We show that f is a morphism. Clearly $f_P(c_0) = m_0$, consider then $(X, t) \in E$, then $f_T((X, t)) = t$, and we have to prove that $f_P^{op} : {}^\bullet t \to {}^\bullet(X, t)$ is a total mapping, which is the case as $X \subseteq B$, $\hat{\mathsf{co}}(X)$ and ${}^\bullet t = f_P(X)$. With a similar argument we have that also $f_P^{op} : t^\bullet \to (X, t)^\bullet$ is a total mapping, furthermore it is easy to see that f_P on ${}^\bullet(X, t)$ and $(X, t)^\bullet$ is a bijection. Consider now $p = f_P((p, \mathcal{T}_p^i))$. Then also $f_T : {}^\bullet(p, \mathcal{T}_p^i) \to {}^\bullet p$ is trivially a total mapping, and the same for $f_T : (p, \mathcal{T}_p^i)^\bullet \to p^\bullet$. The parsimony requirement for f to be a folding morphism is trivially verified. □

A trellis-process is not only a multi-clock net, it is also an unravel net.

Proposition 6.16. *Let* $\mathsf{N} = (N, \nu)$ *be a multi-clock net. Let* $\mathcal{U} = (\mathsf{U}, f)$*, with* $\mathsf{U} = (U, \nu_U)$*, be its trellis process. Then* $U = \langle B, E, W, c_0\rangle$ *is an unravel net.*

Proof. It follows from the fact that it is a trellis process. In fact in [16] it is shown that each state $X \in \mathsf{St}(N)$ is such that N_X is acyclic and each place in this subnet has at most one incoming and one outgoing arc. □

A trellis process can be viewed as a spread net over a suitable ticking domain.

Consider a multi-clock net $\mathsf{N} = (N, \nu)$ of cardinality $v(\mathsf{N}) = n$, where $N = \langle P, T, F, m_0\rangle$, we can define the languages $\mathcal{L}_k^{T_{p_i}} = \{w \in T_{p_i}^* \mid |w| = k\}$, namely the one where all the words have the same length (namely k). Set $\mathsf{A}_i = T_{p_i}$ for some $p_i \in m_0$. The set of elements $\mathcal{A}$ we are looking for is composed by vector-clocks v such that for each index $1 \leq i \leq n$ there exists a number $k \in \mathbb{N}$ such that $v(i) = \mathcal{L}_k^{\mathsf{A}_i}$. The $\mathsf{0}$ is the vector-clock where all entries are the language with just the empty word $\mathcal{L}_0$. Consider now a set of elements of $\mathcal{A}$. We say that the set V of elements of $\mathcal{A}$ is *t-homogeneous* if the following conditions are satisfied:

- for each $v \in V$ there exists at most one index $1 \leq i \leq n$ such that $v(i) \neq \mathcal{L}_0$, and
- for each $v, v' \in V$, if $v(i) \neq \mathcal{L}_0 \neq v'(i)$ then $v = v'$, and
- let $\overline{v} = (\bigcup_{i\in I} v(i)) \setminus \mathcal{L}_0$ if $\bigcup_{i\in I} v(i) \neq \mathcal{L}_0$ and $\overline{v} = \mathcal{L}_0$ otherwise, for each $v_k \in V$, there exists $w_k \in \overline{v_k}$ such that for each i, j, $proj(w_i, \mathsf{A}_j) = proj(w_j, \mathsf{A}_i)$.

The first condition states that each vector-clock of a t-homogeneous set has at most a language which is different from the one containing just the empty word, the second one assures that the *significant* entries of vector-clock are associated to different indexes. The final one assure each significant language contains at least a word which is compatible with a word of the others. Summing up, a t-homogeneous set of vector-clock is composed by vector-clock where the possible entry different from $\mathcal{L}_0$ appears at different indexes, and each vector-clock has at most one entry which is not $\mathcal{L}_0$. Each operation $\mathsf{op}^i \in \mathsf{Op}$ is defined as follows: it takes the vector-clocks $v_1, \ldots, v_k$ such that $\{v_1, \ldots, v_k\}$ is t-homogeneous and it returns a vector-clock v such that for each $i \in I$ it holds that $v(i) = v_j(i)$ if for some $1 \leq j \leq k$ $v_j(i) \neq \mathcal{L}_0$ and $v(i) = \mathcal{L}_0$ if for all

index $1 \leq j \leq k$ it holds $v_j(i) = \mathcal{L}_0$. The τ in this case acts as follows: it takes an index i, a vector-clock v and a label a, and returns the vector-clock v' where all entries but the *i-th* one are set to $\mathcal{L}_0$ and the *i-th* one is set to $v(i) \cdot \{\mathsf{a}\}$.

We can introduce the notion of ticking domain for a trellis process of a multi-clock net $\mathsf{N} = (N, \nu)$. Recall that with T_i we indicate the transition of the subnet N_i, with $i \in \nu(m_0)$.

Definition 6.17. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net of cardinality $v(\mathsf{N}) = n$, where $N = \langle P, T, F, m_0 \rangle$. The ticking domain for the trellis process of N is the quadruple $(\mathcal{A}, \mathsf{Op}, \tau, T)$ where*

- $\mathcal{A} = \{v \mid \forall i \leq n.\ \exists k \in \mathbb{N}.\ v(i) = \mathcal{L}_k^{T_i}\}$, *with* $\mathsf{0} = (\mathcal{L}_0, \ldots, \mathcal{L}_0)$,
- $\mathsf{Op} = \{\mathsf{op}^i \colon \mathcal{A}^i \to \mathcal{A} \mid 1 \leq i \leq n\}$ *where* $\mathsf{op}^i(v_1, \ldots, v_i)$ *is defined when* $V = \{v_1, \ldots, v_i\}$ *is t-homogeneous and* $\mathsf{op}^k(v_1, \ldots, v_i) = v'$ *such that, for each* $1 \leq k \leq v(\mathsf{N})$ $v'(k) = \mathcal{L}_0$ *if for all* $1 \leq j \leq i$. $v_i(k) = \mathcal{L}_0$ *and* $v'(k) = v_j(k)$ *for the unique* v_j *such that* $v_j(k) \neq \mathcal{L}_0$, *and*
- $\tau : I \times \mathcal{A} \times T \to \mathcal{A}$, *takes and index* $i \in I$, *a vector-clock* $v \in \mathcal{A}$ *and an element* $t \in T$, *and returns the vector-clock* v' *where* $\forall j \neq i.\ v'(j) = \mathcal{L}_0$ *and* $v'(i) = v(i) \cdot \{t\}$,

and it is denoted with $\mathcal{D}_{\mathsf{N}}^{tr}$.

We need a further notion. As $\mathcal{U} = (\mathsf{U}, f)$, with $\mathsf{U} = (U, \nu_U)$, is a trellis process, then $U = \langle B, E, W, c_0 \rangle$ is an unravel net, and in particular, for each $b \in c_0$, the subnet U_b is acyclic and each condition b' in U_b has a precise distance from b. This distance is the number of conditions preceding b', which is the same in all paths from b to b'. We denote this distance with $\mathsf{dist}(b')$. Consider now all the paths from the condition $b \in c_0$ to the condition b' belonging to the same partition. These are indeed firing sequences of the subnet U_b, and hence can define the language $\{w \mid \exists \sigma \in \mathsf{FS}_{\{b\}}^{U_b}.\ lead(\sigma) = \{b'\}\ and\ run(\sigma) = w\}$ of all the words associated to the paths from b to b' which we denote $\mathcal{L}_{b'}^{U_b}$

Proposition 6.18. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net, where $N = \langle P, T, F, m_0 \rangle$. Let $\mathcal{U} = (\mathsf{U}, f)$ be its trellis-process, where $\mathsf{U} = (U, \nu_U)$. Then $\mathsf{U}_{\mathcal{D}_{\mathsf{N}}^{tr}, h}$ where*

- $\mathsf{U} = (U, \nu_U)$ *is a multi-clock net where $U = \langle B, E, W, c_0, f_T \rangle$ and ν_U are as in Definition 6.14,*
- $h \colon B \to \mathcal{A}$ *is defined as $h(b) = v$ where $v(i) = \mathcal{L}_b^{U_{\nu_U(b)}}$ when $i = \nu_U(b)$ and $v(i) = \mathcal{L}_0$ otherwise,*

is a spread net over $\mathcal{D}_{\mathsf{N}}^{tr}$.

Proof. The annotations of the conditions in the initial marking are clearly $\mathsf{0}$. The other requirements are easily proved once that one shows that, given an event e in U, $\{h(b) \mid b \in {}^\bullet e\}$ is t-homogeneous and that if $e \in E_i$, then for the unique condition $b' \in e^\bullet$ such that $\nu_U(b') = b_i \in c_0$, $h(b') = v$ where $v(i) \neq \mathcal{L}_0$ and $\mathsf{dist}(b') = \mathsf{dist}(b) + 1$.

Assume that $\{h(b) \mid b \in {}^\bullet e\}$ is not t-homogeneous, hence for some b' and b'' in ${}^\bullet e$ we have that, considering $v' = h(b')$ and $v'' = h(b'')$, $v'(\nu_U(b'))$ is a language which does not contain any word which is *compatible* with a word in the language $v''(\nu_U(b''))$. But ${}^\bullet e$ is a $\hat{\mathsf{co}}$ set of conditions, and each condition is annotated with all the local computation of a fixed length marking the conditions in ${}^\bullet e$, which contradict the assumption, hence $\{h(b) \mid b \in {}^\bullet e\}$ is t-homogeneous.

Clearly the op^i with $i = |\{h(b) \mid b \in {}^\bullet e\}|$ is well defined and applied to $(h(b^1), \ldots, h(b^j))$ returns the vector-clock v, and $\tau(i, v, f_T(e))$ concatenates the symbol $f_T(e)$ to $v(i)$, hence for the condition in $b' \in e^\bullet$ such that $\nu_U(b') = \nu_U(b^i)$ with $b^i \in {}^\bullet e$ it holds that $\mathsf{dist}(b') = \mathsf{dist}(b^i) + 1$. □

Example 6.19. *Consider the multi-clock net in the Example 5.2. In Figure 8 we show the initial part of its trellis process, with the annotations on the conditions of the net that make it a spread net. The condition b_3 is $(p_4, \{t_2\})$ and it has as incoming arcs all the arcs coming from all the events labeled with t_2 caused by p_1, as the language of the component identified by the place p_2 of all the word of length 1 contains just $\{t_2\}$.*

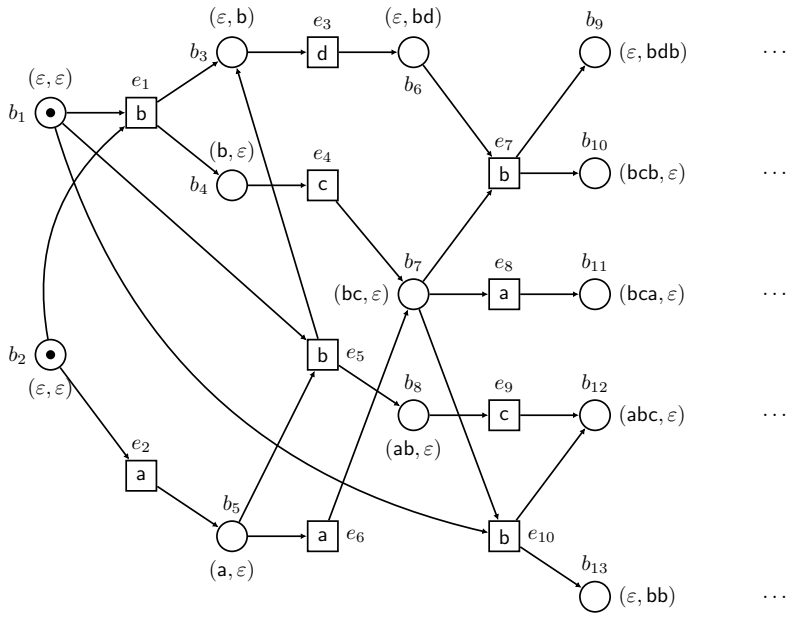


Figure 8: The spread net obtained by the initial part of the trellis process of Figure 5

Remark 6.20. *The Proposition 4.16 and Proposition 4.17 show that, given an information domain, one can construct a spread net annotated with the elements of this domain. This can be done also in the case of ticking domain, along the same lines. Indeed this is implied by the trivial observations that the net of a branching process is an occurrence net, and the net of a trellis process is an unravel one. Furthermore the ticking domains considered are regular and revealing and the multi-clock obtained are properly spread.*

7. Spreading Multi-clock nets

In this section we develop an algorithm to *spread* a multi-clock net once that a ticking domain $\mathcal{D}$ has been selected. Clearly the ticking domain is induced by the net to be spread, and it drives the spreading.

This section is organized as follows: we make precise what the spreading of net is, in analogy to what an unfolding of a net is, then we show how to *construct* an adequate ticking domain $\mathcal{D}$ (Section 7.1) and how to spread a multi-clock net (Section 7.2) with respect to $\mathcal{D}$. Finally, in Section 7.3, we show that indeed the spreading algorithm produces the *best* spread net.

We start by formalizing what is the spreading S of a multi-clock net N with respect to a domain $\mathcal{D}$.

Definition 7.1. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net, with $N = \langle P, T, F, m_0 \rangle$, and let $\mathcal{D}$ be a ticking domain over T, and let $\mathsf{S} = (S, \nu_S)_{\mathcal{D},h}$ be a spread net over $\mathcal{D}$, where $S = \langle P_S, T_S, F_S, m_{0,S}.\ell \rangle$ and $\ell : T_S \to T$ is the labeling mapping. We say that S is a* spreading *of N if*

- *there exists a folding morphism $f : S \to N$, and*
- *for each $C = \{p^1, \ldots, p^i\} \subseteq P_S$ subset of places, such that*
  - *there exists a transition $t \in T$ with ${}^\bullet t = \{f_P(p^1), \ldots, f_P(p^i)\}$, and*
  - *$\mathsf{op}^i(h(p^1), \ldots, h(p^i))$ is defined,*

  then there exists a firing sequence $\sigma \in \mathsf{FS}^S_{m_{0,S}}$ such that $C \subseteq lead(\sigma)$.

The spreading of a multi-clock net N is a spread net on a domain which is constructed starting from the components of the multi-clock net N itself and that is folded onto N. The requirements that a spread net has to satisfy, with respect to the net N, ensure that there is a correspondence between the tuples of annotations for which the operations on the vector-clocks elements are defined and the transitions in the net to be spread. Indeed requiring that whenever the annotations of a subset of places in the spread net, mapped to the preset of a transition t, are a tuple for which the operation in the domain is defined, then these places can be *reached* via a firing sequence, implies that this firing sequence can be extended to another containing a transition of the spread net which will be mapped to t. We recall that in each kind unfolding, if a subset of conditions in the unfolding is mapped to the preset of a transition t in the net, then an event can be added to the unfolding and this event is related to t.

7.1. Defining the ticking domain

Given a multi-clock net $\mathsf{N} = (N, \nu)$, where $N = \langle P, T, F, m_0 \rangle$, we argue on how a domain for the spreading is *constructed*. As N is a multi-clock net, it easy to define an equivalence relation on the words associated to the firing sequences of each component. This determines the entry associated to this component of a generic vector-clock, and this entry will be the equivalence class of a word on the alphabet determined by the transition of the component (and we can use a word in the equivalence class as the representative). Let A be a finite alphabet, let $\sim$ be an equivalence relation defined on A^*, then with $[w]_\sim$ we denote the equivalence class of w with respect to $\sim$, *i.e.* the set $\{w' \in \mathsf{A}^* \mid w \sim w'\}$.

Definition 7.2. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net where $N = \langle P, T, F, m_0 \rangle$ and let $\ell : T \to T$ be the identity labeling mapping. Let $I = m_0$ be a set of indexes and, for each $i \in I$, let $\sim_i$ be an equivalence relation over $\mathsf{Tr}(N_i)$ such that if $w \sim_i w'$ then $w \cdot u \sim_i w' \cdot u$. Let P be a property defined on languages over T. Then $\mathcal{A}$ is the set of vector-clocks v such that for each $i \in I$. $v(i) = [w]_{\sim_i}$ for some $w \in \mathsf{Tr}(N_i)$, and $\mathsf{P}(\{v(i) \mid i \in I\})$ holds.*

The elements of each vector-clock are equivalence classes, and when no confusion arise, we just write $v = (w_1, \ldots, w_n)$ instead of $v = ([w_1]_{\sim_1}, \ldots, [w_n]_{\sim_n})$. The predicate P ensures that the various elements have the proper *shape* (there exists an execution in the component with this shape) and it mixes the visions of each component on the behaviours of the other components. Furthermore P reflects the fact that it must represent a feasible computation of the whole net. In general we assume that, given a vector-clock v, for each $i, j \in I$ (with I the set of indexes of the multi-clock), the words $w_i \in v(i)$ and $w_j \in v(j)$ represent compatible local computations, and the word w in T^* obtained mixing all these words in such a way that $proj(w, T_i) = w_i$, is an execution of the net.

The second ingredient we have to devise concerns the operations in Op. When discussing branching processes and trellis processes we have assumed the existence of a suitable predicate, *e.g.* the *bp-homogeneity* in the case of branching processes. We push this idea a bit further. We need to define when a set of vector-clocks are *homogeneous*, and to this aim we introduce a predicate, which is defined on sets of vector-clocks.

Definition 7.3. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net. Let $\mathcal{A}$ be a set of vector-clocks satisfying P as in Definition 7.2. Let $\mathcal{H}$ be a predicate defined on subsets of vector-clocks. We say that the set $V = \{v_1, \ldots, v_k\}$ is* homogeneous *if $\mathcal{H}(V)$.*

The predicate $\mathcal{H}$ is usually inferred by the firing sequences of each component. For instance one can say that $\mathcal{H}(V)$ holds for the set of vector-clocks such that for each word in $v_i(j)$ there exists a word $w' \in v_k(m)$ and $proj(w, T_m) = proj(w', T_j)$. This predicate captures the minimal idea that for each local computation σ_i in a component there exists a local computation σ_j in another component that is coherent with σ_i. We stress the fact that this is indeed what it is done usually when constructing the branching or the trellis process of a multi-clock net, but it is also what is basically done when constructing the *merged* process of a multi-clock net (see [27]).

The operations are defined for homogeneous sets of vector-clocks, and what they do is to return a set of vector-clocks obtained by suitably combining the words of various languages. To this aim we assume the existence of a partial function g that takes as input a tuple of words and return an homogeneous vector-clock.

Definition 7.4. *Let* $\mathsf{N} = (N, \nu)$ *be a multi-clock net. Let* $\mathcal{A}$ *be a set of vector-clock values as in Definition 7.2 and let* I *be the index set associated to* N*. Let* $g : T_1^* \times \cdots \times T_n^* \rightharpoonup \mathcal{A}$ *be a partial function. For each* $1 \leq i \leq |I|$ *define* $\mathsf{op}^i : \mathcal{A}^i \rightharpoonup \mathbf{2}^{\mathcal{A}}$ *as the mapping taking a* i*-tuple of vector-clocks* $(v_1, \ldots, v_i)$ *such that* $\{v_1, \ldots, v_i\}$ *is homogeneous, and returning the set vector-clock values defined as* $\{v \mid \forall i, j.\ \exists w_i \in v_j(i) \ \wedge \ g(w_1, \ldots, w_n) = v\}$*.*

The intuition here that op^i takes a tuple of local computations (that are compatible as the associated vector-clock is homogeneous) and returns a set of vector-clocks, where each vector-clock is induced by a specific tuple of words using the mapping g. A regular domain is a domain where, given a set if vector-clocks that are homogeneous, the g mapping returns always just one value.

The final ingredient of a ticking domain is the τ which implements the *updating* of the piece of information. A vector-clock v contains for each component an equivalence class of words, and then the ticking mapping just concatenates the symbol to the words in the equivalence class, when the concatenation is defined, *i.e.* the symbol belongs to the alphabet.

Definition 7.5. *Let* $\mathsf{N} = (N, \nu)$ *be a multi-clock net, where* $N = \langle P, T, F, m_0 \rangle$*, and let* $\sim_p$ *be a an equivalence relations over words in* T_p *for each* $p \in m_0$ *Let* P *be a predicate over tuple of languages, and* $\mathcal{H}$ *be a predicate on vector-clocks in* $\mathcal{A}$ *induced by* $\sim_p$ *and* P*. Let* g *be a partial mapping from* $T_1^* \times \cdots \times T_n^*$ *to* $\mathcal{A}$*. Then* $\mathcal{D}_{\mathsf{N}}^{\mathsf{P},\mathcal{H}}$ *induced by* P*,* $\mathcal{H}$ *and* g *is the quadruple* $(\mathcal{A}, \mathsf{Op}, \tau, T)$ *where* $\mathcal{A}$ *is defined according to Definition 7.2,* Op *is defined according to Definition 7.4 and* τ *implement the chosen updating mapping, is a ticking domain for* N*.*

To illustrate how domains are obtained we review briefly some of examples of the previous section and we provide a new one.

Example 7.6. *For the 1-unfolding the property is the one devised in the previous section, namely that just one entry of the vector-clock is not the empty word, and furthermore the only significant entries can be just the counting one. The equivalence relations are trivial: for each component every word is equivalent to the empty word, and for the counting one each word is equivalent to itself. The other ingredients can be easily retrieved. Also the* $\mathcal{H}$ *predicate is obvious, as well as the* g *mapping.*

In the case of the branching process of a multi-clock net, the property is the one stating that a vector-clock $(w_1, \ldots, w_n)$ *is such that* $\forall i, j \in \{1, \ldots, n\}$*.* $proj(w_i, \mathsf{A}_j) = proj(w_j, \mathsf{A}_i)$*. The operations are defined if the tuple of vector-clocks are* bp-homogeneous *(thus for each component there is a maximum) and the result is given by the maximum applied component-wise. The ticking function just concatenate a label to the proper entry of the vector-clock.*

Example 7.7. *Consider the multi-clock net* (N, ν) *in Figure 9. The net has three components, one defined*

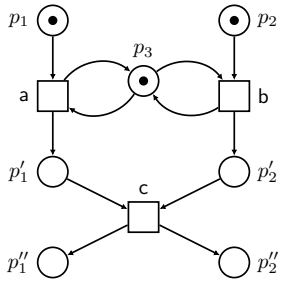


Figure 9: A multi-clock net with three components where the order of execution may play a rôle

by $\nu(p_1) = \nu(p_1') = \nu(p_1'') = p_1$ *(the first one),* $\nu(p_2) = \nu(p_2') = \nu(p_2'') = p_2$ *(the second one) and* $\nu(p_3) = p_3$ *(the third one), and the equivalence on the languages induced by the alphabets* $\{t_1, t_3\}$ *(the one of the first component),* $\{t_2, t_3\}$ *(the one of the second component) and* $\{t_1, t_2\}$ *(the one of the third component) are the following: in the first component and the second component every word is equivalent to* ε*, in the third*

the equivalence is defined as follows: $t_1 \sim t_2t_1$, $t_2 \sim t_1t_2$, $t_1t_1 \sim t_2t_2$ *and for each* $w \in \{t_1, t_2\}^*$ *we have that* $t_1t_1w \sim t_1t_1$ *and* $t_2t_2w \sim t_2t_2$. *The predicate* P *over the vector-clocks induced by this domain is the trivial one: every vector-clock satisfy the property. Observe that there are just four vector-clocks:* $v_0 = (\varepsilon, \varepsilon, \varepsilon) = \mathsf{0}$, $v_1 = (\varepsilon, t_1, \varepsilon)$, $v_2 = (\varepsilon, t_2, \varepsilon)$ *and* $v_3 = (\varepsilon, t_1t_1, \varepsilon)$, *thus the domain is finite. The* $\mathcal{H}$ *is defined for all of them and the operations* op^i *are as follows* $\mathsf{op}^2(v_0, v_0) = \mathsf{op}^2(v_1, v_0) = \mathsf{op}^2(v_0, v_2) = \{v_0\}$, $\mathsf{op}^2(v_1, v_2) = \mathsf{op}^2(v_2, v_1) = \{v_1, v_2\}$ *which implies that the domain is not regular, finally* $\tau(v_0, t_1) = v_1$, $\tau(v_0, t_2) = v_2$ $\tau(v_1, t_2) = v_2$ *and* $\tau(v_2, t_1) = v_1$. *The fact that, for instance,* $\mathsf{op}^2(v_2, v_1)$ *gives* $\{v_1, v_2\}$ *depends from the fact that the piece of information conveyed by this domain is that the annotations depend on the order on which transitions* t_1 *and* t_2 *are performed. Now the choice of which of the possible results in the set* $\{v_1, v_2\}$ *has to be taken depend by the* g *mapping which is defined as* $g(t_1, t_1t_2, t_2) = v_2$ *and* $g(t_1, t_2t_1, t_2) = v_1$.

7.2. Spreading Algorithm

We now present an algorithm to spread a multi-clock net. We recall briefly how an unfolding algorithm works. In the initialization phase a set of conditions corresponding to the initial marking is created and it is related to the initial marking via a folding morphism. Then, for each subset of conditions reachable via a firing sequence that is mapped (via the folding morphism) to the preset of a transition t, a new event labeled t is added to the unfolding, if not present, together with conditions produced by this event and related to the postset of the transition t. This step is iterated and this procedure either terminates when no new event can be added or, if it does not terminate, produces at each step a part of the unfolding, the limit of which is the unfolding itself.

To adapt this procedure to the spreading, we need a notion that is the *semantic* counterpart of the more classical notion of *concurrent* places used in unfoldings.

Definition 7.8. *Given a multi-clock net* $\mathsf{N} = (N, \nu)$ *where* $N = \langle P, T, F, m_0 \rangle$, *we say that* $C \subseteq P$ *is* free of conflict *if there exists a firing sequence* $\sigma \in \mathsf{FS}^N$ *such that* $C \subseteq lead(\sigma)$.

We are now ready to present the algorithm to spread a net with respect to a specific ticking domain, constructed as we have previously devised, which is shown in Figure 10. In the algorithm we use $\|(w_1, \ldots, w_n)$, where each w_i is a word on an alphabet A_i, to denote the set of words on $\bigcup_{i=1}^{n} \mathsf{A}_i$ obtained *shuffling* the words w_i, namely $proj(w, \mathsf{A}_i) = w_i$ for each i and $w \in \|(w_1, \ldots, w_n)$. We make just few comments on this algorithm, as it is the adaptation to this setting of the general algorithm to *unfold* a net. Indeed the main differences are that we do not only check whether a subset of conditions are reached using a firing sequence, but also that the operation op^i of the domain is defined. The outcome of the algorithm is a finite net only in the case where the domain contains a finite number of vector-clock values, otherwise it produce a spread net and a folding morphism at each iteration.

Theorem 7.9. *Let* $\mathsf{N} = (N, \nu)$, *where* $N = \langle P, T, F, m_0 \rangle$, *be a multi-clock net and let* $\mathcal{D}$ *be a domain over* T. *Let* $\mathcal{S} = (\mathsf{S}_{\mathcal{D},h}, f)$ *be the result of the spreading algorithm in Figure 10. Then* S *is a spread net and* $f : S \to N$ *is a folding morphism.*

Proof. Clearly $\mathsf{S}_{\mathcal{D},h}$ is a spread net by construction. The annotation of the condition in the initial marking is the $\mathsf{0}$ of the domain, and for each set of conditions $C \subseteq B$ which is free of conflict, namely a reachable subset in the net S, we have that if there is an event e such that ${}^\bullet e = C$ then $\mathsf{op}^i(h(C))$ is defined, and the annotation of the conditions in the postset is obtained by the τ applied to $f_T(e)$ and a vector-clock in $\mathsf{op}^i(h(C))$. The requirement that the annotation is uniform on the various events in the preset of a condition is guaranteed by construction, as we add a new condition mapped on the same place only if the annotation calculated for the condition is not present. Thus all the requirements of Definition 4.1 are fulfilled. Also f is a folding morphism, as indeed f_P and f_T are total mappings and fulfill the conditions posed on folding morphisms, as by construction, for each event e in E, f_T define a bijection on ${}^\bullet e$ and $e^\bullet$. The parsimony requirement is again satisfied by construction. □

This result has as a consequence that the spreading algorithm applied to a multi-clock net N with respect to a domain $\mathcal{D}$ gives a net which is the spreading in the sense of Definition 7.1.

Input: A multi-clock net $\mathsf{N} = (N, \nu)$ where $N = \langle P, T, F, m_0 \rangle$ and a ticking domain $\mathcal{D} = (\mathcal{A}, \mathsf{Op}\ \tau, T)$ based on P, $\mathcal{H}$ and g.

Output: At each step a spread net $\mathsf{S} = (S, \nu_S)_{\mathcal{D},h}$, where $S = \langle B, E, W, C_0, \ell \rangle$, and a folding mapping f from S to N.

Initialization step: Create $|m_0|$ conditions for S, called C_0 and define a bijection $f_P \colon C_0 \to m_0$. Set $\nu_S(b) = b$ Define, for each $b \in C_0$, $h(b) = \mathsf{0}$, and set $S = \langle C_0, \emptyset, \emptyset, C_0, \ell \rangle$ where $\ell : \emptyset \to T$ is the obvious function. Finally set f as (f_P, f_T) where again $f_T : \emptyset \to T$ is the obvious mapping.

Recursion: Consider the spread net constructed so far $\mathsf{S}_{\mathcal{D},h}$, where $\mathsf{S} = (S, \nu_S)$, with $S = \langle B, E, W, C_0, \ell \rangle$, and the mapping $f : S \to N$.
Let $C = \{b_1, \ldots, b_n\} \subseteq B$ be a non empty subset of conditions such that

- $\forall b, b' \in C.\ \nu_S(b) = \nu_S(b')$ implies $b = b'$,
- $\mathsf{op}^{|C|}(\{h(b) \mid b \in C\})$ is defined,
- C is free of conflict, and
- there exists a transition $t \in T$ such that ${}^\bullet t = f_P(C)$.

Check if E contains an event e such that ${}^\bullet e = C$ and $f_T(e) = t$. If yes consider another subset C' of B. If not then, consider the element $v \in \mathsf{op}^{|C|}(\{h(b) \mid b \in C\})$ and it is such that $g(w_1, \ldots, w_{|C_0|}) = v$, $w \in \|(w_1, \ldots, w_{|C_0|})$ and there exists a firing sequence $\sigma \in \mathsf{FS}^S$ such that $b \in \mathit{lead}(\sigma)$ and $\mathsf{tr}(\sigma) = w = f_T(\mathit{run}(\sigma))$, introduce an event $e \notin E$ and

- add e to E and set $\ell'(e) = T$,
- for each $b_i \in C$ add to W the pair (b'_i, e),
- set $f_T(e) = t$,
- for each index $1 \leq i \leq |C|$ check if there is a condition $b'_i \in B$ such that $h(b'_i) = \tau(i, v, t)$ and $f_P(b'_i) = p_i$ with $p_i \in t^\bullet$, and
  - if yes then add (e, b'_i) to W,
  - if not then add a new condition b''_i to B, add (e, b''_i) to W, set $h(b''_i) = \tau(i, v, t)$, $\nu_S(b''_i) = \nu_S(b_i)$ and $f_P(b''_i) = p_i$
- set $\nu_O(p') = \phi_P^{-1}(\nu_O(p))$

Let B' be the set of the new added conditions, E' be the set of added events and W' be the added arcs, ν'_S be the extension of ν_S to the new added places, then $\mathsf{S} = (\langle B \cup B', E \cup E', W \cup W', C_0 \rangle, \nu'_S, \ell')\mathcal{D}h'$ is the resulting spread net and f the folding morphism.

Figure 10: The spreading algorithm

Corollary 7.10. *Let* N *be a multi-clock net and let* $\mathcal{D}$*. Let* $\mathcal{S} = (\mathsf{S}_{\mathcal{D},h}, f)$ *be the result of the spreading algorithm inFigure 10. Then* S *is a spreading of* N*.*

Example 7.11. *Consider the multi-clock net in Figure 11 (which is depicted together with its two components).*

The elements of the ticking-domain are obtained according to the following two equivalence relations. The alphabet for the first component is $\mathsf{A}_1 = \{s, t, u, v, z\}$ *and the equivalence relation* $\sim_1$ *is induced by the following equations:* $\varepsilon = \varepsilon$, $s = t$, $su = tu$, $suz = su$, $tuz = su$, $sus = s$, $tus = s$, $sut = s$, $tut = s$, $suvu = su$, $tuvu = su$, $u = \varepsilon$, $v = \varepsilon$, $ss = s$, $ts = s$ *and* $tt = t$ *and* $w = \varepsilon$ *for each* $w \in \mathsf{A}_1^*$ *such that* $|w| \geq 4$ *and* $w \neq suvu$ *and* $tuvu$*. The equivalence classes obtained are then* $[\varepsilon]_{\sim_1}$, $[s]_{\sim_1}$, $[su]_{\sim_1}$, $[suv]_{\sim_1}$ *and* $[suz]_{\sim_1}$*. The alphabet is* $\mathsf{A}_2 = \{u, w, z\}$ *and the equivalence relation* $\sim_2$ *is based on the following equations:* $uwu = uz$, $\varepsilon = \varepsilon$ *and for each other word* $w \in \mathsf{A}_2^*$ *beside the ones involved in these equations, we have* $w = \varepsilon$*. 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*

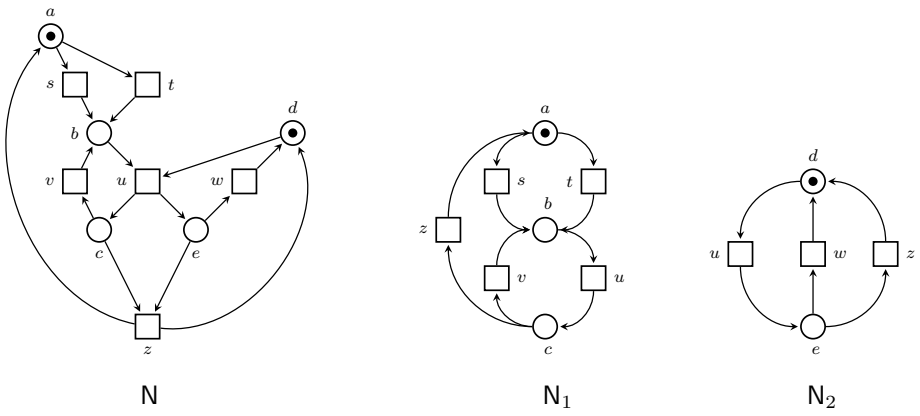


Figure 11: A multi clock net and its component

classes are identified with their representative. Finally the property P *they have to satisfy is that they are compatible, namely that for each element in the first component there must be an element in the second component and such that both agree on common symbols.*

The predicate $\mathcal{H}$ *stipulate that* v *and* v' *are homogeneous whenever* $\exists w \in v(1)$ *and* $w' \in v(2)$ *such that* $proj(w, \mathsf{A}_{1,2}) = proj(w', \mathsf{A}_{1,2})$*. The operation* op^1 *just return the same vector-clock, the operation* $\mathsf{op}^2(v, v')$ *returns the vector-clock* $(v(1), v'(2))$*. Finally the* τ *concatenates the symbol to the proper entry of the vector-clock.*

In the figure places are annotated with the pair (a, v) *where* a *is the place in the net* N *and* v *is a vector-clock. The result of the spreading is in Figure 12.*

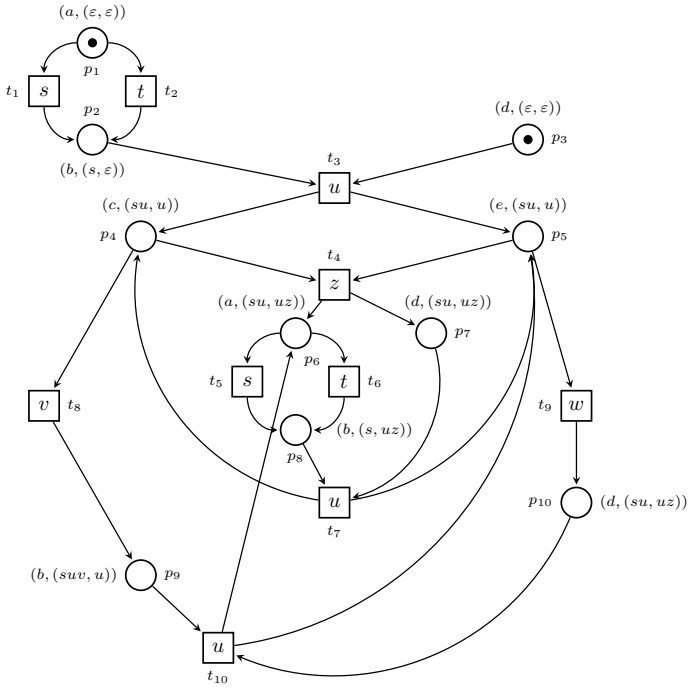


Figure 12: The spreading of the net in Figure 11

The domains considered so far have been always regular domains. The next example discuss the construction in the case the domain is not a regular one.

Example 7.12. *In Figure 13 we show the spreading of the multi-clock net in Example 7.7. The domain and the operation are those devised in the Example 7.7. Each condition is decorated with the annotation (v_i), its internal name b_k and to which place in the multi-clock net is mapped to, and the same for the events, where the label is inside the event itself. Observe that b_3 and b_8 are annotated like b_6 and b_9, but the execution of e_5 and e_6 gives different annotations, as e_5 depends on the fact that in the net N the transition b is executed first and e_6 depends on the fact that in the net N the transition a is executed first.*

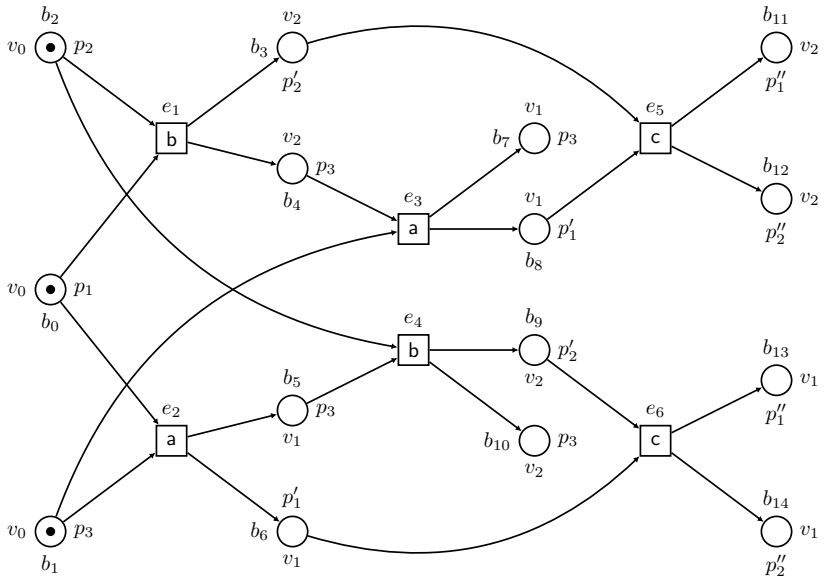


Figure 13: The spreading on the net in Figure 9 with the respect to the domain in Example 7.7

7.3. Relating spread nets

Given two spread nets S and S' over the same domain $\mathcal{D}$, namely $\mathsf{S} = (S, \nu)_{\mathcal{D},h}$ and $\mathsf{S}' = (S', \nu')_{\mathcal{D},h'}$, we relate them. The idea is that there is a morphism $f : S \to S'$ such that if $(p, p') \in f_P$ then $h(p) = h'(p')$.

This idea is formalized in the definition below.

Definition 7.13. *Let $\mathsf{S} = (S, \nu)_{\mathcal{D},h}$ and $\mathsf{S}' = (S', \nu')_{\mathcal{D},h'}$, be two spread nets on the same domain $\mathcal{D}$. A morphism $f : S \to S'$ is* annotations preserving *if for each $(p, p') \in f_P$ it holds that $h(p) = h'(p')$*

With the aid of this notion we can show the following result, which basically state that the spreading algorithm gives us the best result possible, as all the other spreadings are related to this one, when the domain is regular.

Theorem 7.14. *Let $\mathsf{N} = (N, \nu)$ be a multi-clock net and let $\mathcal{D}_{\mathsf{N}}$ be the associated domain (with respect to P, $\mathcal{H}$ and g). Assume that $\mathcal{D}_{\mathsf{N}}$ is regular. Let $\mathcal{S} = (\mathsf{S}_{\mathcal{D}_{\mathsf{N}},h}, f)$ be the spreading of N with respect to $\mathcal{D}_{\mathsf{N}}$, and let $\mathsf{S}'_{\mathcal{D}_{\mathsf{N}},h'}$ be another spread net with respect to $\mathcal{D}_{\mathsf{N}}$ such that there exists a morphism $g : S' \to N$. Then there exists a unique morphism $h : S' \to S$ which is annotations preserving.*

Proof. Consider a spread net $\mathsf{S}'_{\mathcal{D}_{\mathsf{N}},h'}$ where $\mathsf{S}' = (S', \nu_{S'})$ and $S' = \langle B', E', W', C'_0 \rangle$, being the labeling induced by the folding morphism, and assume that there exists a folding morphism $g : S' \to N$. As each event in a spread net can be executed, we can associate to each event e a *depth* which is the length of the shortest firing sequence leading to it. Formally $\mathsf{depth}(e) = \min(\{|run(\sigma)| \mid \sigma \in \mathsf{FS}^{S'}. \sigma\,[e\rangle\})$, and

this is well defined. We can define the subnets S'_i of S' as follows: the events E'_i are the set of events $\{e \in E' \mid \mathsf{depth}(e) \leq i\}$, the conditions are B'_i is $C'_0 \cup {}^\bullet E'_i \cup E'^\bullet_i$, the initial marking is C'_0 and the flow relation W'_i is the restriction of W' to the conditions and events in B'_i and E'_i respectively. The information mapping h'_i is the restriction of h to the places in B'_i.

We now construct the unique mapping k from S' to S as follows. First, for each $b' \in C'_0$, we set $k_P(b') = b$ such that $b \in C_0$ and $f_P(b) = g_P(b')$. This define the morphism k^0. The morphism is obviously annotation preserving as $\mathsf{S}'_{\mathcal{D}_\mathsf{N},h'}$ is a spread net with respect to $\mathcal{D}_\mathsf{N}$. Now consider the events in E' which have depth equal to 0. These are events that are enabled at the initial marking C'_0. Consider the event e' such that ${}^\bullet e' \subseteq C'_0$. In the spreading of N we have that the event e corresponds to the transition $g_T(e')$, and then we set $h^1_T(e') = e$ and to each condition in the postset of e' we associate the condition in the postset of e belonging to the same component, and we do this for all the events in E' with depth equal to 0. In this way we define k^1. Iterating the procedure we construct, for each $> 0i$, a morphism k^i from S'_{i-1} to S. Each of the morphism is, by construction, annotation preserving. The limit of this construction gives the desired morphism.

We have to show that this is unique. Assume there is another one, say l. Being a different morphism, and being S' a spread net, there should be an event, say e' at depth n, which is mapped by k to e_1 and by l at another e_2. But this is possible only if the domain is non regular, as in this case there could be a choice of the possible events associated in the spreading to the transition $g_T(e')$. Hence the thesis. □

8. Discussion

On constructing domains. The results in the previous section show that spreading a net can give a more compact representation of the behaviour of net with respect to many approaches. Using the notion of domain is possible to introduce much more different criteria with respect to the one introduced in literature to *compress* these structures. The key ingredient for the spreading of a multi-clock net is clearly the notion of ticking domain. We have seen abstractly how to construct a ticking domain for a multi-clock net. We discuss on how to do this in a more concrete and uniform way, by providing some examples and foreseeing some techniques.

Consider the multi-clock net $\mathsf{N} = (N, \nu)$ where $N = \langle P, T, F, m_0 \rangle$, and consider the automaton N_p with $p \in m_0$. The equivalence relation on which each element of the ticking domain is defined is clearly an equivalence relation defined on the language $\mathcal{L}_{N_p} = \{run(\sigma) \mid \sigma \in \mathsf{FS}^{N_p}_{\{p\}}\}$, and we can look at a uniform way to define this equivalence relation when this equivalence relation is of finite index. We do not see this as a limitation as we are introducing spread nets to have a finite and compact representation of the behaviours of a net. To this aim we can imagine that the equivalence relation we are interested in may quite well be represented as equivalence relation induced by a finite state automaton. Consider then a finite state automaton $D_p = (Q, T_p, \delta, q_0, F)$, the equivalence on two words $w, w' \in \mathcal{L}_{N_p}$ is then defined as $\hat{\delta}(q_0, w) = \hat{\delta}(q_0, w')$ where $\hat{\delta}$ is the extension on words of the transition function $\delta : Q \times T_p \to Q$ which is defined classically.

The Figure 14 illustrates some possibilities. For simplicity we identify the domain with the finite state automaton which define the desired equivalence relation. Consider the net automaton N_1 with just two transitions a and b in Figure 14. For this net automaton we can put forward some finite state automata defining equivalences on the words on the alphabet $\{a.b\}$. The automaton D_1 conveys the idea that all the computations containing at leas one symbol of the net automaton N are to be considered as equivalent, the D_2 that all the computations have to be considered equivalent, the D_3 has three equivalence classes, depending on the number of a (zero, one or two), the third has four equivalence classes, one contains just the empty word, the second the words where a and b alternates, the starting symbol is a and the final one is b, the D_3 the words where a and b alternates, the starting symbol is b and the final one is a, and the D_4 equates all the words where there are at least two adjacent equal symbols. The figure shows their spreading (in the spreading we omit to indicate to which place the conditions are mapped as it is obvious). For the net automaton N_2 we consider the finite state automaton D_5 in . The spreading is the spread net S_5 in the same figure.

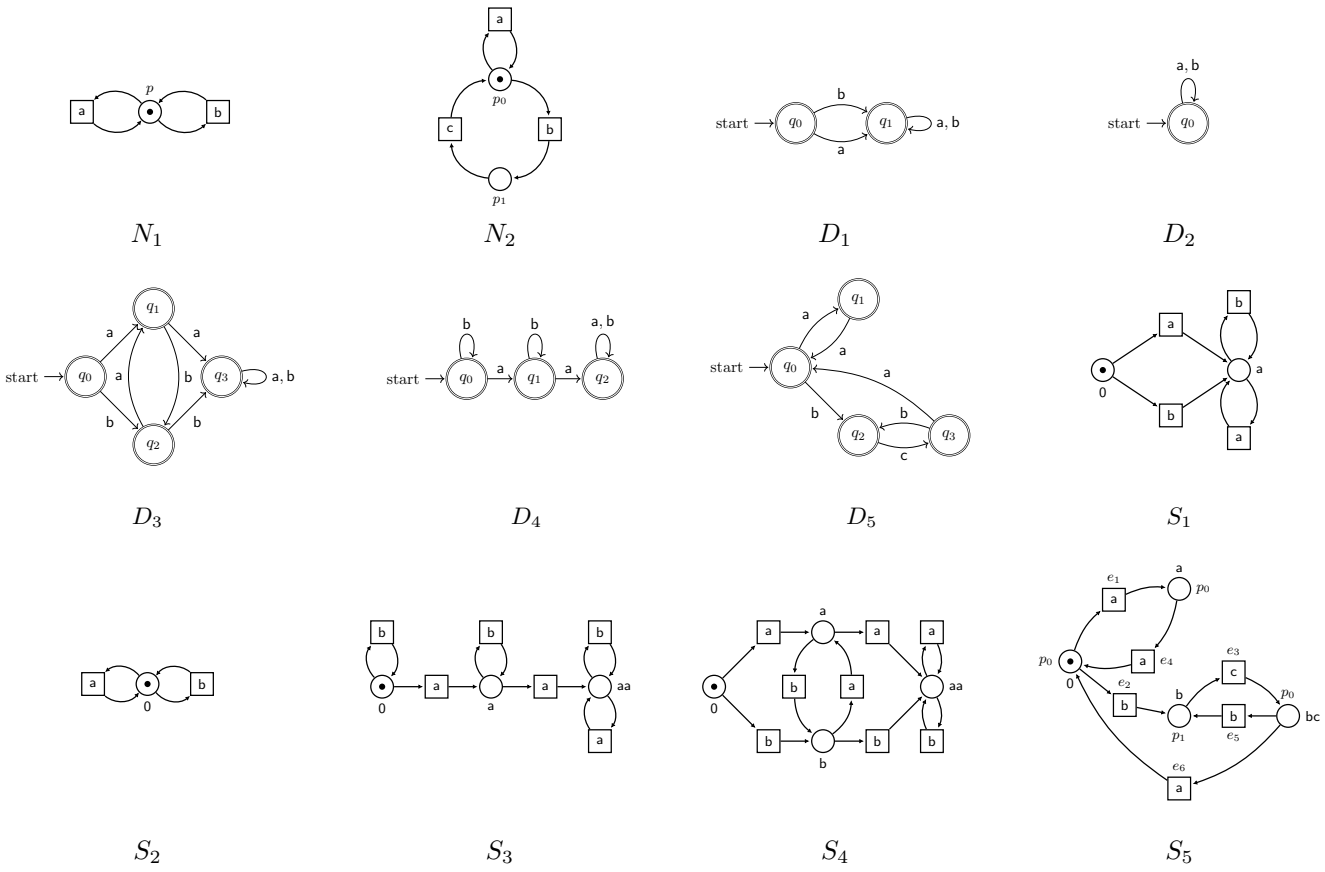


Figure 14: Two net automata N_1 and N_2, five finite state automata for equivalence relations on words over $\{\mathsf{a}, \mathsf{b}\}$ (D_1, D_2, D_3 and D_4), a finite state automaton for an equivalence relation on words in $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ (D_5), the spreading S_i of N_1 with respect to the domains D_i, $1 \leq i \leq 4$ and the spreading S_5 of N_2 with respect to the domain D_5

The previous example suggests that many of the interesting equivalences where also time is folded are easily represented as a finite automata.

An issue is now on how to *introduce* a domain that is able to characterize the fact that each transition is observably executed. We can then imagine that, given a component N_p of a multi-clock net, it is possible to associate a finite state automaton representing this. Given a net automaton $N = \langle P, T, F, \{p\}\rangle$, the automaton can be constructed as follows: states are triple (p', i, t) where $p' \in P$, i is the *distance* of p' from p, and $t \in T$ is a transition putting a token in p' or $\perp$ if p' is p. The distance is the shortest path from p to p'. Consider again the finite state automaton D_5 in Figure 14. This is obtained by the net automaton N_2 as follows: q_0 is $(p_0, 0, \perp)$, q_1 is $(p_0, 0, a)$, q_2 is $(p_1, 1, b)$ and q_3 is $(p_0, 0, c)$. Clearly the distances in this net automaton are 0 and 1. The transitions with an incoming arc in the place p_0 are a and c, whereas the only one with an incoming arc in the place p_1 is b. Thus the only possible obtainable states are q_0, q_1, q_2 and q_3. The δ is defined almost accordingly to the firing mapping '$[\cdot\rangle$' of the net automaton. We stipulate that $\delta((p, i, t), t')$ is either equal to $(p', i + 1, t')$ if there exists a state $(p', i + 1, t')$ and $\{p\}\,[t'\rangle\,\{p'\}$ or the state (p, j, x) for $j < i$, with x equal to t' or $\perp$ and there exists a path in the automaton from (p, j, x) to (p, i, t). In the spreading S_5 the events folding the time are e_4, e_5 and e_6.

Consider again the multi-clock net in Figure 9, but now consider the domains induced by the finite state automata in Figure 15. The elements of the ticking domain are constructed using the same predicate we defined previously. The vector-clocks are $v_0 = (\varepsilon, \varepsilon, \varepsilon)$, $v_1 = (\mathsf{a}, \mathsf{a}, \varepsilon)$, $v_2 = (\varepsilon, \mathsf{b}, \mathsf{b})$, $v_3 = (\mathsf{a}, \varepsilon, \mathsf{b})$ and $v_4 = (\mathsf{ac}, \varepsilon, \mathsf{bc})$. The operations on the domain are defined as follows: $\mathsf{op}^2(v_0, v_0) = v_0$, $\mathsf{op}^2(v_0, v_1) = v_1$, $\mathsf{op}^2(v_0, v_2) = v_2$ and $\mathsf{op}^2(v_1, v_3) = v_3 = \mathsf{op}^2(v_2, v_3)$. $\tau(v_0, \mathsf{a}) = v_1$, $\tau(v_0, \mathsf{b}) = v_2$, $\tau(v_1, \mathsf{b}) = v_3$, $\tau(v_2, \mathsf{a}) = v_3$

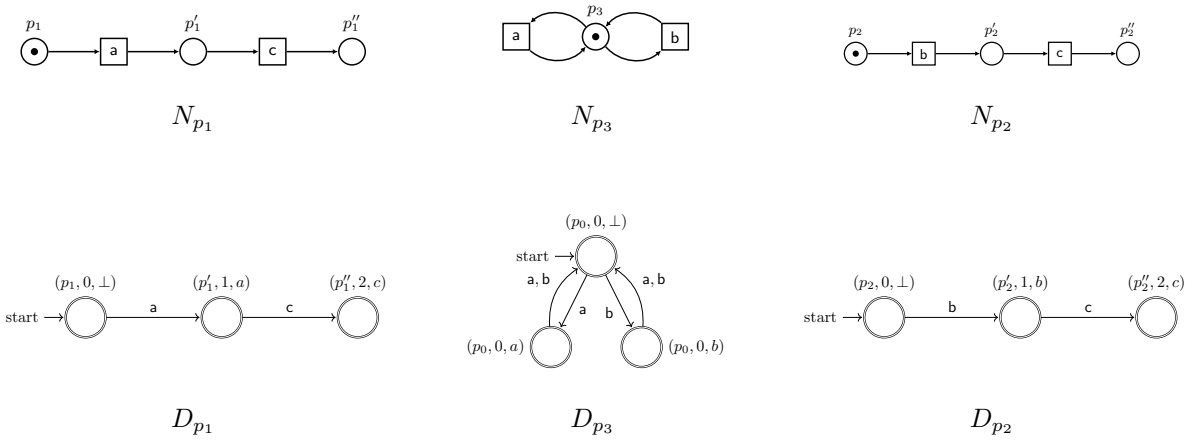


Figure 15: The three components of the multi-clock net in Figure 9 and the finite state automata associated to each of them.

and $\tau(v_3, \mathsf{c}) = v_4$. The spreading with respect to this domain is the one depicted in Figure 16. Observe that there is no any longer a confusion, as the domain is now regular.

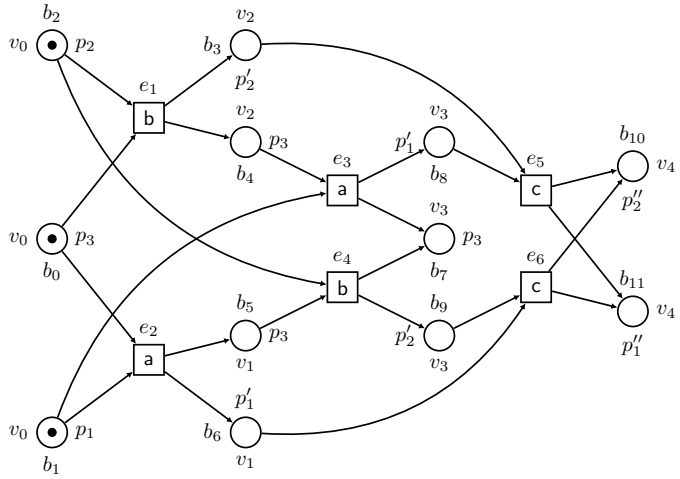


Figure 16: The spreading of the net in Figure 9 with respect to the domain induced by the automata for the local prefixes

Merged processes. In Section 6 we have presented various kinds of unfolding and discussed how they can be seen as spread nets on suitable domains. The common characteristic of these unfolding is that there exists a direct representation. The notion of *merged* process has been left out, but also merged processes can be seen as spread nets.

We recall how a merged process is defined. Starting from a branching process, the conflicting conditions that satisfy the requirements

- are equally labeled (*i.e.* mapped to the same place of the net to be unfolded), and
- in the past of each of them the number of conditions bearing the same label of the *final* one is the same

can be *merged*, that means that they are identified. Consider the multi-clock net in Figure 5 and the part of its branching process depicted in Figure 7. There the conditions b_4, b_9 and b_{17} can be identified as they

are mapped on the place p_3 and in their past there is no condition with the same label. If we look at the annotations of these conditions, it is easy to see that just one occurrence of the transition b has been executed and being this transition the only one putting a token in the place p_3, all the local execution in the left component where the transition b has fired just once have to be considered equivalent. The elements of the ticking domain are then vectors of languages where, for each place in each component, the language contains all the traces associated to the firing sequences of the component putting in the place the i-th token. The property of the vectors-clock is the usual one, each of the language must contain an execution which is compatible with one of the other components.

The following figure show the initial part of the merged process obtained by the initial part of the branching process.

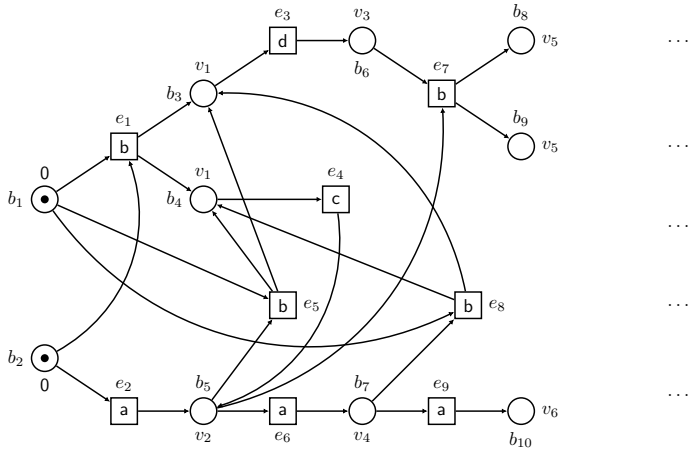


Figure 17: The initial part of the merged process of the net in Figure 5

The annotations are the vectors of languages, that here are represented as regular expressions. $\mathsf{0}$ is $(\underline{\varepsilon}, \underline{\varepsilon})$, v_1 is $(\mathsf{a}^*\mathsf{b}, \mathsf{b})$, v_2 is $(\mathsf{a} + \mathsf{bc}, \mathsf{b} + \underline{\varepsilon})$, v_3 is $(\mathsf{a}^*\mathsf{b}, \mathsf{bd})$, v_4 is $((\mathsf{a} + \mathsf{bc})(\mathsf{a} + \mathsf{bc}), \mathsf{bd} + \mathsf{b} + \underline{\varepsilon})$, v_5 is $((\mathsf{a}^* + \mathsf{bc})\mathsf{b}, \mathsf{bdb})$ and v_6 is $((\mathsf{a} + \mathsf{bc})(\mathsf{a} + \mathsf{bc})(\mathsf{a} + \mathsf{bc}), \mathsf{bdbd} + \mathsf{bd} + \mathsf{b} + \underline{\varepsilon})$ are those annotating the conditions shown in the Figure 17. For a discussion on how to obtain these annotations we refer to [28].

Composing spread nets and information domains. Multi-clock nets come equipped with a clear operation on how these can be composed: the multi-clock net $\mathsf{N} = (N, \nu)$ resulting from the composition of two multi-clock nets $\mathsf{N}_1 = (N_1, \nu_1)$ and $\mathsf{N}_2 = (N_2, \nu)$ has as places the union of the places (that are disjoint), as transitions the union of the transitions (that may be not disjoint) and the other components are defined accordingly in such a way that restricting to the places and transitions of a N_i ones get exactly N_i. Similarly one may imagine a way of composing ticking domains provided that the property the vector-clock of the compound ticking domain is known, which is normally the case. The new elements of the ticking domain are just vectors that are obtained by those of the two ticking domains and that satisfy the property, and the other operations are easily definable. It is then clear that the composition of the two spread nets that are the spreading of N_1 and N_2 (spreading with respect the proper ticking domains) will give the spread net which is the spreading of N with respect to the compound ticking domain.

On a categorical setting. 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 and we have shown that under the assumption that the domain is regular, the spreading construction can be considered the *best* one. The above consideration suggests that a categorical treatment is possible.

One should start stating what is a multi-clock net over a domain hence not necessarily a spread one. Let $\mathsf{N} = (N, \nu)$ be a multi-clock net where $N = \langle P, T, F, m_0 \rangle$, and $\mathcal{D} = (\mathcal{A}, \mathsf{Op}, \tau, \mathsf{Lab})$ be a domain. Then $\mathcal{N} = \mathsf{N}_{\mathcal{D},h}$, where $h : P \to \mathcal{A}$, is a multi-clock net over a domain $\mathcal{D}$. With respect to the notion of spread

net, the annotations on places are not subjected to any constraints, hence a net over a domain is simply a net where places are annotated with elements belonging to the set $\mathcal{A}$.

The notion of mapping relating domains can be the following one. Let $\mathcal{D} = (\mathcal{A}, \mathsf{Op}, \tau, \mathsf{Lab})$ and $\mathcal{D}' = (\mathcal{A}', \mathsf{Op}', \tau', \mathsf{Lab}')$. Then a *domain mapping* is the triple $\phi = (\phi_{lab}, \phi_{dom}, \phi_{op})$ where $\phi_{lab} : \mathsf{Lab} \to \mathsf{Lab}'$, $\phi_{dom} : \mathcal{A} \to \mathcal{A}'$ and $\phi_{op} : \mathsf{Op} \to \mathsf{Op}'$ are (partial) mappings such that

- $\phi_{dom}(0_{\mathcal{A}}) = 0_{\mathcal{A}'}$,
- for all $V \subseteq \mathcal{A}$ and for all $\mathsf{op}^i \in \mathsf{Op}$, if $\mathsf{op}^i(V)$ is defined and equal to $V' \subseteq \mathcal{A}$, then also $\phi_{op}(\mathsf{op}^i)(\phi_{dom}(V))$ is defined and $\phi_{of}(\mathsf{op}^i)(\phi_{dom}(V)) = \phi_{dom}(V')$, and
- $\phi_{dom}(\tau(i, v, \mathsf{a})) = \tau'(i, \phi_{dom}(v), \phi_{lab}(\mathsf{a}))$.

The requirements we put on the mappings between domains is that the 0 of the first domain is mapped onto the 0 of the second domain, and that the operations are preserved. Clearly domain mappings compose.

We have to specialize the notion of morphism on multi-clock nets in order to take into account domains. Recall that, given two multi-clock nets $\mathsf{N} = (N, \nu)$ and $\mathsf{N}' = (N', \nu')$, a morphism $f : \mathsf{N} \to \mathsf{N}'$ is net morphism such that the partitions are preserved. Let $\mathcal{N} = \mathsf{N}_{\mathcal{D},h}$ and $\mathcal{N}' = \mathsf{N}'_{\mathcal{D}',h'}$ be two nets over a domain. Then the pair $\mathsf{f} = (f, \phi)$ is a morphism whenever f is a net morphism, ϕ is a domain mapping and $\phi_{dom}(h(p)) = h'(p')$ for all p' such that $(p, p') \in f_P$.

It is then easy to see that nets over a domain and morphisms between them form a category, called **DomNet**. It is enough to show that nets morphisms compose, as well as domains mappings. Let $\mathsf{f} = (f, \phi) : \mathcal{N} \to \mathcal{N}'$ and $\mathsf{f}' = (f', \phi') : \mathcal{N}' \to \mathcal{N}''$ be two morphisms. Then $\mathsf{f} \circ \mathsf{f}' = (f \circ f', \phi \circ \phi') : \mathcal{N} \to \mathcal{N}''$ is a well defined morphism and $\phi_{dom} \circ \phi'_{dom}(h(p))$ is $\phi'_{dom}(\phi_{dom}(h(p))$ and, as f is a morphism, for each p' such that $(p, p') \in f_P$ we have that $\phi_{dom}(h(p)) = h'(p')$. Now also f' is a morphism, hence for each p'' such that $(p', p'') \in f'_P$ we have $\phi'_{dom}(h'(p')) = h''(p'')$, but then also for each $(p, p'') \in f_P \circ f'_P$ we have that $\phi'_{dom}(\phi_{dom}(h(p)) = h''(p'')$. The subcategory of spread net over a domain is the category where objects are spread nets and it is called **Spread**. With this treatment, we have that the Theorem 7.14 account to say that the spreading define a universal arrow.

9. Conclusions and future works

In this paper we have proposed the notion of spread net as the net capable of representing the different net based semantics presented in literature. The various unfoldings can be seen as spreadings with respect to specific information domains. We have then applied the notion to multi-clock nets, where the behaviour of the whole net can be obtained by the behaviours of the components. 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 piece of information depends on two elements. One element is the piece of 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.

Spread nets may easily *fold* conflicts which account to consider as equivalent computations representing conflicting alternatives, as it is done in trellises processes or in merged processes, but, differently from these approaches, they may fold as well time, thus qualifying for a finite representation of possible infinite behaviours. 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 (see [29] for the various approaches presented to unfold a net with respect a token interpretation [30]).

The spreading of multi-clock nets has the advantage of being able to unfold each component according to a criterion, and this criterion is not necessarily the same one for all the components, as it is in the existing approaches. Thus we allow that certain components of the net are *executed* according the individual token philosophy (meaning the the whole history is preserved) or the collective token philosophy (meaning that all the histories are equated) or other different interpretations, where some computations are equated depending

on suitable criteria. Clearly what is developed for multi-clock nets may as well be applied to any kind of net where a clear understanding of what the components are and how they do interact.

In this paper we have considered either information domains tuned to express the various kinds of unfolding, or very simple ones, without making any real consideration on the kind of properties one would like to express on the system whose behaviours is described by the spread net. In a sense the target of the paper has been the setting of a *framework* to express the semantics of nets, based on spread nets and the spreading of a net, hence we have not focussed on the properties an information domain can or should express. Indeed the main advantage of the framework proposed is the fact that it that the act of spreading a net is somehow independent on the chosen information domain, which is a parameter of the algorithm, and then it can be used in quite different contexts. The choice of the information domains influences the properties one would like to characterize. We can then imagine to find information domains which characterize *prefixes* of a net, or information domains enforcing some other property. We can also imagine to have a way to *synthesize* information domains starting form the properties one would like to observe, and some suggestions are indeed present in Section 7 and Section 8, but these are just a basis for a theory of information domains. We stress again that the kind of semantics represented in particular spreading depends on the information domain chosen and hence on the properties one would like to focus on.

We have not yet investigated an interesting issue, namely what is the brand of event structure related to spread net, like it is done in [24]. 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.

The categorical treatment of spread nets has to be further investigated. In the previous section we have put forward some ideas. We are confident that this is the correct direction, and the gain of setting the approach in a categorical framework will give us constructions like products for free. Certainly once fixed the kind of information domain, the categorical treatment follows the usual lines, with some advantages, for instance the pieces of information can be used to sort out which transitions have to be synchronized in two spread nets.

Acknowledgment. We would like to thank the anonymous reviewers for their useful criticisms and suggestions that have helped us to greatly improve the paper.

References

[1] E. Smith, W. Reisig, The semantics of a net is a net, in: K. Voss, H. J. Genrich, G. Rozenberg (Eds.), Concurrency and Nets, Springer Verlag, 1987, pp. 461–479.

[2] U. Goltz, W. Reisig, The non-sequential behavior of Petri nets, Information and Control 57 (2/3) (1983) 125–147.

[3] M. Nielsen, G. D. Plotkin, G. Winskel, Petri Nets, Event Structures and Domains, Part 1, Theoretical Computer Science 13 (1981) 85–108.

[4] G. Winskel, Event Structures, in: W. Brauer, W. Reisig, G. Rozenberg (Eds.), 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, Vol. 255 of Lecture Notes in Computer Science, Springer Verlag, 1987, pp. 325–392.

[5] J. Engelfriet, Branching processes of Petri nets, Acta Informatica 28 (6) (1991) 575–591.

[6] P. Degano, J. Meseguer, U. Montanari, 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, 1989, pp. 175–185.

[7] P. Degano, J. Meseguer, U. Montanari, Axiomatizing the algebra of net computations and processes, AI 33 (7) (1996) 641–667.

[8] K. L. McMillan, Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits, in: G. von Bochmann, D. K. Probst (Eds.), Computer Aided Verification, Fourth International Workshop, CAV '92, Montreal, Canada, June 29 - July 1, 1992, Proceedings, Vol. 663 of Lecture Notes in Computer Science, Springer, 1993, pp. 164–177.

[9] G. Casu, G. M. Pinna, Petri nets and dynamic causality for service-oriented computations, in: A. Seffah, B. Penzenstadler, C. Alves, X. Peng (Eds.), SAC 2017 Conference Proceedings, ACM, 2017, pp. 1326–1333.

[10] G. Casu, G. M. Pinna, Merging relations: A way to compact Petri Nets' behaviors uniformly, in: F. Drewes, C. Martín-Vide, B. Truthe (Eds.), LATA 2017 Conference Proceedings, Vol. 10168 of Lecture Notes in Computer Science, 2017, pp. 325–337.

[11] S. Balaguer, T. Chatain, S. Haar, Building occurrence nets from reveals relations, Fundamamenta Informaticae 123 (3) (2013) 245–272.

[12] S. Haar, C. Kern, S. Schwoon, Computing the reveals relation in occurrence nets, Theoretical Computer Science 493 (2013) 66–79.
[13] J. Esparza, S. Römer, W. Vogler, An Improvement of McMillan's Unfolding Algorithm, Formal Methods in System Design 20 (3) (2002) 285–310.
[14] V. Khomenko, M. Koutny, W. Vogler, Canonical prefixes of Petri net unfoldings, Acta Informatica 40 (2) (2003) 95–118.
[15] V. Khomenko, A. Kondratyev, M. Koutny, W. Vogler, Merged Processes: a new condensed representation of Petri net behaviour, Acta Informatica 43 (5) (2006) 307–330.
[16] E. Fabre, Trellis processes: A compact representation for runs of concurrent systems, Discrete Event Dynamic Systems 17 (3) (2007) 267–306.
[17] E. Fabre, G. M. Pinna, Toward a uniform approach to the unfolding of nets, in: M. Bartoletti, S. Knight (Eds.), Proceedings 11th Interaction and Concurrency Experience, ICE 2018, Madrid, Spain, June 20-21, 2018., Vol. 279 of EPTCS, 2018, pp. 21–36.
[18] A. W. Mazurkiewicz, Trace theory, in: W. Brauer, W. Reisig, G. Rozenberg (Eds.), 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, Vol. 255 of Lecture Notes in Computer Science, Springer Verlag, 1987, pp. 297–324.
[19] S. Abramsky, A. Jung, Domain theory, in: S. Abramsky, D. M. Gabbay, T. S. E. Maibaum (Eds.), Handbook of Logic in Computer Science, Vol. III, Oxford University Press, 1994, pp. 1–168.
[20] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. W. Mislove, D. S. Scott, Continuous Lattices and Domains, Vol. 93 of Encyclopedia of Mathemtics and its Applications, Cambridge University Press, 2003.
[21] G. Boudol, Flow Event Structures and Flow Nets, in: Semantics of System of Concurrent Processes, Vol. 469 of Lecture Notes in Computer Science, Springer Verlag, 1990, pp. 62–95.
[22] P. Baldan, A. Corradini, U. Montanari, Contextual Petri nets, asymmetric event structures and processes, Information and Computation 171 (1) (2001) 1–49.
[23] A. Corradini, N. Busi, A. Corradini, G. M. Pinna, Domain and event structure semantics for Petri nets with read and inhibitor arcs, Theoretical Computer Science 323 (1-3) (2004) 129–189.
[24] R. J. van Glabbeek, G. D. Plotkin, Configuration structures, event structures and Petri nets, Theoretical Computer Science 410 (41) (2009) 4111–4159.
[25] R. Langerak, Bundle Event Structures: A Non-Interleaving Semantics for Lotos, in: M. Diaz, R. Groz (Eds.), Formal Description Techniques, V, Proceedings of the IFIP TC6/WG6.1 Fifth International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, FORTE '92, Perros-Guirec, France, 13-16 October 1992, Vol. C-10 of IFIP Transactions, North-Holland, 1992, pp. 331–346.
[26] R. J. van Glabbeek, G. D. Plotkin, Configuration structures, in: D. Kozen (Ed.), Proceedings of 10^{th} Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, 1995, pp. 199–209.
[27] V. Khomenko, A. Mokhov, Direct construction of complete merged processes, The Computer Journal 57 (5) (2014) 693–707.
[28] G. Casu, G. M. Pinna, Flow unfolding of multi-clock nets, in: G. Ciardo, E. Kindler (Eds.), Petri Nets 2014 Conference Proceedings, Vol. 8489 of Lecture Notes in Computer Science, Springer, 2014, pp. 170–189.
[29] G. M. Pinna, How much is worth to remember? a taxonomy based on Petri Nets Unfoldings, in: L. M. Kristensen, L. Petrucci (Eds.), Petri Nets 2011 Conference Proceedings, Vol. 6709 of Lecture Notes in Computer Science, Springer Verlag, 2011, pp. 109–128.
[30] R. J. van Glabbeek, The individual and collective token interpretation of Petri nets, in: M. Abadi, L. de Alfaro (Eds.), Proceedings of CONCUR 2005, Vol. 3653 of Lecture Notes in Computer Science, Springer, 2005, pp. 323–337.