Flow unfolding of multi-clock nets*

Giovanni Casu and G. Michele Pinna

Dipartimento di Matematica e Informatica, Università di Cagliari, Cagliari, Italy
giovanni.casu@unica.it, gmpinna@unica.it

Abstract. Unfoldings of nets are often related to event structures: each execution of a net can be viewed as a configuration in the associated event structure. This allows for a clear characterization of dependencies and the conflicts between occurrences of transitions in the net. This relation is somehow lost if more compact representations of the executions of nets are considered, *e.g.* in trellises or merged processes of multi-clock nets. In this paper we introduce an unfolding, called *flow unfolding*, that turns out to be related to flow event structures, hence dependencies and conflict are still represented. Furthermore, this unfolding gives also a more compact representation of the executions of a multi-clock net, similarly to what approaches like trellises or merged processes do.

1 Introduction

In recent years various new approaches have been proposed to unfold a Petri net. Unfoldings are meant to give a representation of the computations of a Petri net which can be used for several purposes, and among others we recall verification of properties [1], diagnosis of systems [2] and obviously modeling of systems computations [3]. The classical notion of unfolding as developed by Winskel in [3] and Engelfriet in [4] suffers of the state explosion problem, as each event in the unfolding has a unique history, thus possibly equivalent computations leading to the *same* state in the original net are kept distinct. To overcome this problem two main approaches have been pursued. One focuses in finding a *finite* representation of an unfolding, *e.g.* with the notion of prefix [5]. The other tries to identifies computations under suitable equivalences. To the latter approach *trellises* and *merged processes* can be ascribed. Indeed, with different motivations, both Fabre in [6] with his *trellises* and Khomenko, Kondratyev, Koutny, and Vogler in [7] with their *merged processes* have proposed two different ways to reduce dramatically the size of the unfolding of a safe net. In the first proposal a safe net is considered as the product of finite state automata (hence turned into a so called *multi-clock* net), and the unfolding of all the components are then glued together by merging conditions which have the same heights (the height of a condition being its history in the proper automata) and identifying transitions representing correct synchronizations among the various automata; whereas in the second proposal conditions representing the same occurrence of

* Work partially supported by Aut. Region of Sardinia under grants LR 7/07 CRP-17285 (TRICS), PIA 2010 "Social Glue", by MIUR PRIN 2010-11 "Security Horizons"

a token are identified and consequently transitions bearing the same label and having the same preset and postset are merged.

The classical notion of unfolding has a clear and useful connection with another central notion of concurrency theory, namely the one of *event structure* [3]. Indeed, to the classical unfolding of a safe net it is possible to associate a *prime event structure* and vice versa. Dependencies and conflicts among events are represented faithfully in prime event structure, but some prices have to be paid: possibly equivalent computations may be not *equate*, thus producing again the explosion of the states space.

To be able to relate unfoldings to event based models a minimal requirement should be verified: in the unfolding a transition can be executed only once in a run (see [8], where *configuration structures* are introduced). It is worth to stress that in some event based models dependencies and conflicts are not directly available, but have to be deduced (*e.g.* in [8] or [9]). We observe that there is no clear relation between trellises or merged processes and the more used notions of event structures (where dependencies and conflicts are represented). For merged process, even when restricted to multi-clock nets, there is no relation at all, as they do not fulfil the minimal requirement stated above, namely that each transition is executed only once in a run.

Boudol in [10] and with Castellani in [11] are among the first in proposing another notion of event structure, where an event may have several histories, namely *flow* event structures. Other variations of the basic concept of event structures have been proposed subsequently, with various purposes which we will not investigate here. We briefly recall among others: asymmetric event structure of Baldan, Corradini and Montanari [12], or inhibitor event structure of Baldan, Busi, Corradini and Pinna [13], bundle event structure of Langerak [14] or the one associated to the Muller's unfolding of Gunawardena [15]. The configurations of flow event structures and those of prime event structures ordered by inclusion form a prime algebraic domains. The notion of configuration in prime event structure is extremely natural, and modeling the causal relation as a partial order and stating the inheritance principle for the conflict relation (along the causal relation) have played a major rôle in the success of prime event structures, as they imply that each event has a unique history. In this respect flow event structures are less manageable. On the one hand the dependency relation and the conflict relation are required to be irreflexive the first one and symmetric the second one, but no clear interaction among them is foreseen. On the other hand the notion of configuration is much more complicated with respect to the one of prime event structure because it is on this level that a choice should be done among the various histories of an event: one which is compatible with the histories of the other events have to be selected. We observe that prime event structures can be seen as special cases of any other kind of event structures.

In this paper we propose a notion, flow unravel net, where a dependency relation and a conflict relation are easily defined, but many equivalent runs may be equated, similarly to merged processes or trellises. This is obtained by adding suitable *control* places. The notion of flow unravel net is a proper extension of

that of causal net, as each causal net can be turned into a flow unravel net. Control places are used to define a relation of dependency among transitions of the net: a transition depends on another if in the preset of the transition there is a control place that is in the postset of the other. A conflict relation is instead defined stating an *immediate conflict* among two transitions that share an *internal* place *i.e.* a place which is not a control one, and that represents a shared resource, and this conflict is inherited along the dependency up to a certain point, which is identified as the point where two alternative histories determine the same future evolution.

Flow unravel nets are the basis to develop the notion of flow unfolding. We, similarly to what does Fabre in [6], unfold a multi-clock net, as the information provided by the various components of a multi-clock net eases the construction. The unfolding algorithm of Esparza, Römer and Vogler in [21], without cutoffs, can be easily adapted to our purposes, once it is understood when, in the construction, two elements are the same (both for places and transitions). In [16] some notions of equivalence among transitions based of the neighborhoods of transitions have been proposed. Here we adapt these equivalence to our purposes: two transitions are equivalent if they have compatible histories and have the same *future* evolution. The result of the unfolding algorithm of a multi-clock net N is a flow unravel net and a mapping that folds the flow unravel net onto N. It is worth to observe that each safe net can be turned into a multi-clock net having the same *behaviour*, *i.e.* the same firing sequences and with a (step) case graph isomorphic to the one of the original safe net.

The paper is organized as follows: in the next section we recall all the needed definitions concerning nets and we define what an unravel net and a flow unravel net are. In Section 3 we show how prime event structures and classical unfoldings are related as well as flow unravel nets and flow event structures. In Section 4 we present our unfolding algorithm and we show that indeed the algorithm gives a flow unravel net (hence a flow event structure can be associated to it). In Section 5 we compare our construction with trellises and merged process. Some conclusions are drawn in the final section.

2 Nets

Notations: With $\mathbb{N}$ we denote the set of natural numbers and with $\mathbb{N}^+$ the set of natural numbers without zero, i.e., $\mathbb{N} \setminus \{0\}$. Let X be a set, with $|X|$ we denote the cardinality of the set. Let A be a set, a *multiset* of A is a function $m : A \to \mathbb{N}$. The set of multisets of A is denoted by μA. The usual operations on multisets, like multiset union $+$ or multiset difference $-$, are used. We write $m \leq m'$ if $m(a) \leq m'(a)$ for all $a \in A$. If $m \in \mu A$, we denote by $[\![m]\!]$ the multiset defined as $[\![m]\!](a) = 1$ if $m(a) > 0$ and $[\![m]\!](a) = 0$ otherwise; sometimes we will use $[\![m]\!]$ as the denotation of the subset $\{a \in A \mid [\![m]\!](a) = 1\}$ of A. Finally, when a multiset m of A is a set, *i.e.* $m = [\![m]\!]$, we write $a \in m$ to denote that $m(a) \neq 0$, namely that $a \in [\![m]\!]$, and often confuse the multi set m with the set $\{a \in A \mid m(a) \neq 0\}$.

Given an alphabet Σ, with Σ^* we denote as usual the set of words on Σ with ϵ as the empty word. The length of a word is defined as usual and, with abuse of notation, it is denoted with $|\cdot|$. Given a word w and a subset A of the alphabet, $proj(w, A)$ is the word obtained deleting all occurrences of symbols not belonging to A.

Given a partial order $(D, \sqsubseteq)$, with $\lfloor d \rfloor$ we denote the set $\{d' \in D \mid d' \sqsubseteq d\}$.

Nets: We first review the notions of Petri net and of the token game.

Definition 1. *A* Petri net *is a 4-tuple $N = \langle S, T, F, m\rangle$, where S is a set of* places *and T is a set of* transitions *(with $S \cap T = \emptyset$), $F\colon (S \times T) \cup (T \times S) \to \mathbb{N}$ is the* flow *mapping, and $m \in \mu S$ is called the* initial marking.

Subscripts or superscript on the net name carry over the names of the net components. Given an $x \in T$, with ${}^\bullet x$ we denote the multiset on S defined as ${}^\bullet x(s) = 1$ if $F(s, x)$ and 0 otherwise, and with $x^\bullet$ we denote the multiset on S defined as $x^\bullet(s) = 1$ if $F(x, s)$ and 0 otherwise. Similarly, given an $y \in S$, with ${}^\bullet y$ and $y^\bullet$ we denote the multisets on T defined respectively as ${}^\bullet y(t) = 1$ if $F(y, t)$ and 0 otherwise, and $x^\bullet(t) = 1$ if $F(t, y)$ and 0 otherwise. For $x \in S \cup T$, ${}^\bullet x$ and $x^\bullet$ are called the *preset* and *postset* respectively of x. Given a finite multiset of transitions $A \in \mu T$ we write ${}^\bullet A$ for $\sum_{t \in T} A(t) \cdot {}^\bullet t$ and $A^\bullet$ for $\sum_{t \in T} A(t) \cdot t^\bullet$.

A net $\langle S, T, F, m\rangle$ 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 s to a transition t iff $F(s, t) > 0$ and an arc connects a transition t to a place s iff $F(t, s) > 0$. We assume that all nets we consider are such that $\forall t \in T$ ${}^\bullet t$ and $t^\bullet$ are not empty.

A finite multiset of transitions A is enabled at a marking m, if m contains the pre-set of A. Formally, a finite multiset $A \in \mu T$ is *enabled* at m if ${}^\bullet A \leq m$. In this case, to indicate that the execution of A in m produces the new marking $m' = m - {}^\bullet A + A^\bullet$ we write $m\,[A\rangle\, m'$. Steps and firing sequences, as well as reachable markings, are defined in the usual way. The set of reachable markings of a net N is denoted with $\mathcal{M}_N$. Each reachable marking can be obviously reached with a firing sequence where just a transition is executed at each step. Given a firing sequence $m\,[A_1\rangle\, m_1 \cdots m_{n-1}\,[A_n\rangle\, m_n$, we say that m_n it *reached* by $\sum_{i=1}^{n} A_i$.

A net is said *safe* whenever its places hold at most one token in all possible evolutions. Formally:

Definition 2. *A net $N = \langle S, T, F, m\rangle$ is said* safe *in the case that $F\colon (S \times T) \cup (T \times S) \to \{0, 1\}$ and each marking $m \in \mathcal{M}_N$ is such that $m = [\![m]\!]$.*

Subnet: A subnet of a net is a net obtained restricting places and transitions, and correspondingly also the multirelation F and the initial marking. We can restrict either the transitions or the places.

Definition 3. *Let $N = \langle S, T, F, m\rangle$ be a Petri net and let $T' \subseteq T$. Then the subnet generated by T' is the net $N|_{T'} = \langle S', T', F', m'\rangle$, where $S' = \{s \in$*

$S \mid F(t,s) > 0$ or $F(s,t) > 0$ for $t \in T'\} \cup \{s \in S \mid m(s) > 0\}$, F' is restriction of F to S' and T', and m' is the multiset on S' obtained by m restricting to places in S'.

Analogously we can restrict the net to a subset of places.

Definition 4. *Let $N = \langle S, T, F, m\rangle$ be a Petri net and let $S' \subseteq S$. Then the subnet generated by S' is the net $N|_{S'} = \langle S', T', F', m'\rangle$, where $T' = \{t \in T \mid F(t,s) > 0$ or $F(s,t) > 0$ for $s \in S'\}$, F' is restriction of F to S' and T', and m' is the multiset on S' obtained by m restricting to places in S'.*

Multi-clock nets: Safe nets can be seen as formed by various *sequential* components (automata) synchronizing on common transitions. This intuition is formalized in the notion of *multi-clock* nets, introduced by Fabre in [6].

Definition 5. *A* multi-clock *net N is the pair $(\langle S, T, F, m\rangle, \nu)$ where $\langle S, T, F, m\rangle$ is a safe net and $\nu : S \to [\![m]\!]$ is a mapping such that*

- *for all $s, s' \in [\![m]\!]$, it holds that $s \neq s'$ implies $\nu^{-1}(s) \cap \nu^{-1}(s') = \emptyset$,*
- *$\bigcup_{s \in [\![m]\!]} \nu^{-1}(s) = S$,*
- *$\nu|_{[\![m]\!]}$ is the identity, and*
- *for all $t \in T$. ν is injective on $[\![{}^\bullet t]\!]$ and on $[\![t^\bullet]\!]$, and $\nu([\![{}^\bullet t]\!]) = \nu([\![t^\bullet]\!])$.*

Given $s \in S$, with $\overline{s}$ we denote the subset of places defined by $\nu^{-1}(\nu(s))$. The consequences of the two requirements, namely (a) $\nu|_{[\![m]\!]}$ is the identity and (b) ν is injective on the preset (postset) of each transition and that $\nu([\![{}^\bullet t]\!]) = \nu([\![t^\bullet]\!])$, is that for each $s \in [\![m]\!]$, the net $\langle S, T, F, m\rangle|_{\overline{s}} = \langle \overline{s}, T_{\overline{s}}, F_{\overline{s}}, m_{\overline{s}}\rangle$ is a state-machine net, *i.e.* the preset and the postset of each transition has at most one element. State-machine nets can be considered as finite state automata, and the net $\langle S, T, F, m\rangle$ can be seen as the *union* of the various components. Sometimes multi-clock nets will be identified with the underlying safe net $N = \langle S, T, F, m\rangle$ and the partition mapping will be denoted with $\nu(N)$. It should be stressed that the partition is not unique. Consider the net in figure 1, the two partitions are identified by the following partition mapping $\nu(s) = s$, $\nu(r) = s$, $\nu(p) = p$ and $\nu(q) = p$.

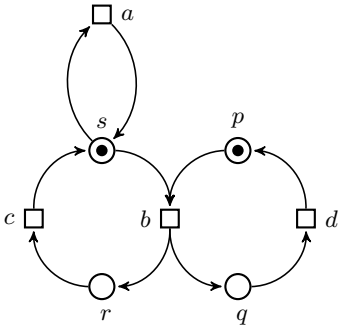


Fig. 1: A multi-clock net.

Occurrence Nets: The notion of occurrence net we introduce here is the one called 1-occurrence net and proposed by van Glabbeek and Plotkin in [17]. First we need to introduce the notion of *state*.

Definition 6. *Let $N = \langle S, T, F, m\rangle$ be a Petri net, a* state *is any finite multiset X of transitions with the property that the function $m_X : S \to \mathbb{Z}$ given by $m_X(s) = m(s) + \sum_{t\in T} X(t) \cdot (t^\bullet(s) - {}^\bullet t(s))$, for all $s \in S$, is a reachable marking of the net which is reached by X. With $\mathcal{X}(N)$ we denote the states of the net N.*

A state contains (in no order) all the occurrences of the transitions that have been fired to reach a marking. Observe that a trace of a net is a suitable linearization of the elements of a state X. On the notion of state the notion of occurrence net is based:

Definition 7. *An* occurrence net *$O = \langle S, T, F, m\rangle$ is a Petri net where each state is a set, i.e. $\forall X \in \mathcal{X}(N)$ it holds that $X = [\![X]\!]$.*

The intuition behind this notion is the following: regardless how tokens are produced or consumed, an occurrence net *guarantees* that each transition can *occur* only once (hence the reason for calling them occurrence nets).

Causal Nets: The notion of causal net we use here is the classical one, though it is often called occurrence net. The different name is due to the other notion of occurrence net we use here. Given a net $N = \langle S, T, F, m\rangle$, we define $s <_N t$ iff $F(s, t)$ and $t <_N s$ iff $F(t, s)$, and $\leq_N$ is the transitive and reflexive closure of this relation. For denoting places and transitions we use B and E (see [18] and [3, 19]) and call them conditions and events respectively. A causal net is essentially an acyclic net equipped with a conflict relation (which is deduced using the relation F, as causal nets are safe nets).

Definition 8. *A* causal net *$C = \langle B, E, F, m\rangle$ is a safe net satisfying the following restrictions:*

- *$\forall b \in m$, ${}^\bullet b = \emptyset$,*
- *$\forall b \in B$. $\exists b' \in m$ such that $b' \leq_C b$,*
- *$\forall b \in B$. ${}^\bullet b$ is either empty or a singleton,*
- *for all $e \in E$ the set $\{e' \in E \mid e' \leq_C e\}$ is finite, and*
- *$\#$ is an irreflexive and symmetric relation defined as follows:*
  - *$e \#_i e'$ iff $e, e' \in E$, $e \neq e'$ and ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$,*
  - *$x \# x'$ iff $\exists y, y' \in E$ such that $y \#_i y'$ and $y \leq_C x$ and $y' \leq_C x'$.*

The intuition behind this notion is the following: each condition b represents the occurrence of a token, which is produced by the *unique* event in ${}^\bullet b$, unless b belongs to the initial marking, and it is used by only one transition (hence if $e, e' \in b^\bullet$, then $e \# e'$). On causal net it is natural to define a notion of *causality* among elements of the net: we say that x is *causally dependent* from y iff $y \leq_C x$. Given a causal net $C = \langle B, E, F, m\rangle$, if $\forall b \in B$ it holds that $b^\bullet$ is a singleton, we say that it is a *conflict-free* causal net. Observe that a conflict-free causal net may be considered as a net representing a non-interleaving execution of a system. The following observation essentially says that each transition (event) in a causal net is executed only once:

Proposition 1. *let C be a causal net, then C is also an occurrence net.*

Unravel Nets: Causal nets capture dependencies (and conflicts) whereas occurrence nets capture the unique occurrence property of each transition. We introduce now a notion of net which will turn to be, so to say, in between occurrence nets and causal nets. Like in the case of occurrence nets we want to assure that each transition happens just once, and similarly to causal nets we want still to be able to retrieve dependencies among the firings of transitions, though in a more *semantical* way, as we require that each state of the net (together with the adjacent arcs) induces a conflict-free causal net.

Definition 9. *An* unravel net $R = \langle S, T, F, m\rangle$ *is an occurrence net such that (a) R is safe, and (b) for each state $X \in \mathcal{X}(R)$ the net $R|_{[\![X]\!]}$ is a conflict-free causal net.*

This notion covers trivially the one of causal net.

Proposition 2. *Let C be a causal net. Then C is an unravel net.*

Flow Unravel Nets: Unravel nets are *locally* acyclic, *i.e.* for each execution of the net the causal dependencies are clear. We would like to have this information also *structurally* available. To this aim we introduce the notion of *flow unravel* nets.

Definition 10. *An unravel net $R = \langle S, T, F, m\rangle$ is a* flow unravel net *iff the set of places S can be divided into two subsets S_i and S_c (*internal *and* control *places respectively) such that*

1. *$S_i \cup S_c = S$ and $S_i \cap S_c = m$,*
2. *$\forall t \in T.\ {}^\bullet t \cap S_x \neq \emptyset$ and $t^\bullet \cap S_x \neq \emptyset$, for $x \in \{i, c\}$,*
3. *$\forall t, t' \in T.\ t^\bullet \cap {}^\bullet t' \cap S_c \neq \emptyset$ implies that $t^\bullet \cap {}^\bullet t' \cap S_i \neq \emptyset$,*
4. *$\forall t, t' \in T.\ t^\bullet \cap {}^\bullet t' \cap S_c \neq \emptyset$ implies that $|t^\bullet \cap {}^\bullet t' \cap S_c| = 1$, and*
5. *(S_c, T, F_c, m_c) is a connected and acyclic net, where F_c and m_c are the restriction of F and m to S_c respectively.*

The idea behind flow unravel nets is to be able to keep track of dependencies among transitions, and this is achieved using the places in S_c. We require that each transition is connected both to internal places (which represent the used or produced resources) and control places (which are used to describe the dependencies). Requirement (4) is a kind of *economicity* criterion: among two transitions there may be at most one control place. Observe that $\mathcal{X}(R) \subseteq \mathcal{X}(R|_{S_i})$ as each transition is connected to internal places. We stress that requiring that among two transitions there may be at most one control place does not imply that the control place must have just one outgoing arc.

Flow unravel nets are conservative extensions of causal net. $C = \langle B, E, F, m\rangle$ is turned into a flow unravel net as follows: for each pair of events e, e' such that $e^\bullet \cap {}^\bullet e' \neq \emptyset$ add a place (e, e'), for each event e such that $\forall e' \in E\ {}^\bullet e' \cap e^\bullet = \emptyset$ add a place $(e, -)$. These added places are the places in $S_c \setminus m$. The flow relation F' is obtained as expected: $F'(e, (e, e')) = 1 = F'((e, e'), e')$ for all $(e, e') \in S_c$ and $F(e, (e, -)) = 1$ for all $(e, -) \in S_c$.

We end this section introducing some auxiliary notions. Assume that there exists a set of labels $\mathcal{A}$ and a labelling function $l\colon S_i \to \mathcal{A}$. Then, for each state of a flow unravel net we can define a mapping that associates to each *internal* place the number of equally labeled internal places preceding it with respect to that state (and this is finite as a state of an unravel net gives a conflict-free causal net).

Definition 11. *Let $R = \langle S, T, F, m\rangle$ be a flow unravel net, and $l : S_i \to \mathcal{A}$ be a labelling function. For each $X \in \mathcal{X}(R)$ call $C_X = (R|_{S_i})|_X$ and S_i^X are the places of this causal net. Then for each $X \in \mathcal{X}(R)$ define $tok^l_X\colon S_i \to \mathbb{N}$ as follows: $tok^l_X(s) = |\{s' \in S_i^X \mid s' \leq_C s \ \wedge \ l(s) = l(s')\}|$.*

Given a flow unravel net $R = \langle S, T, F, m\rangle$, $s \in S_i$ and $X \in \mathcal{X}(R)$, we say that s is *used* in X iff there exists $t \in X$ with $s \in t^\bullet$. A flow unravel net is *uniformly* labelled with respect to a given labelling function iff to each internal place a unique number can be associated, for every state. Formally:

Definition 12. *Let $R = \langle S, T, F, m\rangle$ be a flow unravel net, and $l : S_i \to \mathcal{A}$ be a labelling function. We say that R is* uniformly labelled *with respect to l iff for all $s \in S_i$ and for all $X, X' \in \mathcal{X}(R)$ such that s is used in X and X', then $tok^l_X(s) = tok^l_{X'}(s)$.*

The purpose of the labelling function is to identify token occurrences in the notion of flow unfolding as it will be clear later.

3 Event structures and nets

We recall now some basic notions on *event structures* and their relations with nets.

Prime event structures: Prime event structures (PES) [20, 3] are a simple event-based model of concurrent computations in which events are considered as atomic and instantaneous steps, which can appear only once in a computation. The relationships between events are expressed by two binary relations: *causality* and *conflict*. The relevance of the notion of prime event structure is rooted in the well known relation with another central notion for modeling computations, namely the one of *domain*.

Definition 13. *A* prime event structure (PES) *is a tuple $P = (E, \leq, \#)$, where E is a set of* events *and $\leq$, $\#$ are binary relations on E called* causality relation *and* conflict relation *respectively, such that:*

1. *the relation $\leq$ is a partial order and $\lfloor e \rfloor$ is finite for all $e \in E$, and*
2. *the relation $\#$ is irreflexive, symmetric and hereditary with respect to $\leq$,* i.e.*, $e\#e'$ and $e' \leq e''$ imply $e\#e''$ for all $e, e', e'' \in E$.*

An event can occur only after some other events (its causes) have taken place, and the execution of an event can prevent the execution of other events. This is formalized via the notion of *configuration* of a PES $P = \langle E, \leq, \#\rangle$, which is

a subset of events $C \subseteq E$ such that for all $e, e' \in C$ $\neg(e \# e')$ (*conflict-freeness*) and $\lfloor e \rfloor \subseteq C$ (*left-closedness*).

Causal nets and PES are closely related: let $C = \langle B, E, F, m \rangle$ be a causal net. Then $(E, \leq, \#)$ is a PES, where $\leq$ and $\#$ are the causality and conflict relations obtained by the causal net (see [3]). To a configuration of the associated PES it is possible to associate a marking in the causal net.

Proposition 3. *Let $C = \langle B, E, F, m \rangle$ be a causal net, and let $X \subseteq E$ be a configuration of $(E, \leq, \#)$. Then X is a state of C and $\mathbf{mark}(X) = m_X$ is the marking reached executing the events in X.*

We observe that, given a configuration X of the PES associated to a causal net C, the subnet $C|_X$ is a causal conflict-free net.

Flow event structures: Flow event structures (FES) are another event-based model for concurrent computations [10]. Like prime event structures also flow event structures have a clear relation with prime algebraic domains [11], but in a FES an event may have several histories, whereas in a PES $(E, \leq, \#)$ the history of the event e is simply $\lfloor e \rfloor$, which is a configuration.

Definition 14. *A* flow event structure (FES) *is a tuple $F = (E, \prec, \#)$, where E is a set of* events *and $\prec$, $\#$ are binary relations on E called* precedence relation *and* conflict relation *respectively such that $\prec$ is irreflexive and $\#$ is symmetric.*

With respect to PES, the causality is substituted with a precedence relation and it is stipulated that it is irreflexive, whereas for the conflict relation the inheritance principle is abandoned. Consider a FES $F = (E, \prec, \#)$. With $e \natural e'$ we denote the reflexive closure of $\#$ and, given $X \subseteq E$, with $\leq_X$ the relation $(\prec \cap (X \times X))^*$. A configuration of flow event structure is a subset of events which is conflict-free and where each event e is *justified*: if an event preceding it is not in the configuration there must be another one preceding e. The precedence relation hence cover not only the immediate causality but also the various possible alternatives. Formally we have that X is a configuration iff $(X, \leq_X)$ is a partial order such that $\forall e \in X.\ \{e' \in X \mid e' \leq_X e\}$ is finite, and given any $e \in X$, if $e' \prec e$ and $e' \notin X$ then there exists $e'' \in X$ and $e' \natural e'' \prec e$. In fact obviously $\prec$ is irreflexive and $\#$ is symmetric. Flow unravel nets are good candidates to be related to flow event structures. On a flow unravel net net $R = \langle S, T, F, m \rangle$ we can define the following relations, which can be considered as *dependency* and *conflict* relations:

- $t \prec t'$ iff $t^\bullet \cap {}^\bullet t' \cap S_c \neq \emptyset$,
- $t \#_i t'$ iff ${}^\bullet t \cap {}^\bullet t' \cap S_i \neq \emptyset$,
- $\#\ \subseteq\ T \times T$ is the minimal symmetric and irreflexive relation such that
  - $\#_i\ \subseteq\ \#$, and
  - $t \# t'$ and $t' \prec t''$ and $\neg(t \prec^* t'')$ then $t \# t''$.

Hence, a flow unravel net net has ingredients similar to the ones that a causal net has, namely a causality relation and a conflict relation, the main difference is that the conflict relation in a flow unravel net is only partially inherited. We can prove the following propositions:

Proposition 4. *Let $R = \langle S, T, F, m\rangle$ be a flow unravel net, then $(T, \prec, \#)$ is a flow event structure.*

Proposition 5. *Let $R = \langle S, T, F, m\rangle$ be a flow unravel net, and let X be a state of R. Then X is a configuration of $(T, \prec, \#)$.*

Proof. It is enough to show that, given any $t \in X$, if $t' \prec t$ and $t' \notin X$ then there is a $t'' \in X$ such that $t'' \prec t$ and $t' \# t''$, as the other requirements are met trivially by the definition of state and of unravel net. Assume that there is $t' \prec t$, $t' \notin X$ and there is no $t'' \in X$ which is $t'' \prec t$ and such that $t' \# t''$. But this would violate that $R|_{\lbrack\!\lbrack X \rbrack\!\rbrack}$ is a causal net.

Proposition 6. *Let $R = \langle S, T, F, m\rangle$ be a flow unravel net, and let $X \subseteq T$ be a configuration of $(T, \prec, \#)$. Then X is a state of R and $\mathbf{mark}(X) = m_X$ is the marking reached executing the events in T.*

Proof. By induction on the size of X (the elements of X). If X is the empty set then $\mathbf{mark}(X) = m = m_X$. Assume it holds for X of size n and let us prove for $X \cup \{t\}$, with $t \notin X$. As $X \cup \{t\}$ is a configuration t is a maximum with respect to $\leq_X$ as otherwise X would not be a configuration, hence it remains to prove that $m_X[t\rangle$. Assume it is not, then there must be a place in ${}^\bullet t$ which is not marked in m_X. But this can happen only if there is a transition t' in X which have used the token in this place, which means that $t \# t'$ contradicting that $X \cup \{t\}$ is a configuration.

4 Flow unfolding of a multi-clock net

In this section we construct a flow unravel net which turns out to be an unfolding of a multi clock net.

The unfolding of a net N is usually defined as a pair: a net with certain properties and a mapping that associate this net to N in such a way that to each state of N a state of the unfolding of N correspond and vice versa. We recall the notion of morphism between safe nets [3].

Definition 15. *Let $N = \langle S, T, F, m\rangle$ and $N' = \langle S', T', F', m'\rangle$ be nets. A* morphism *$h : N \to N'$ is a pair $\langle h_T, h_S\rangle$, where $h_T : T \to T'$ is a partial function and $h_S \subseteq S \times S'$ is a relation such that*

- *for each $s' \in m'$ there exists a unique $s \in S$ and $s\ h_S\ s'$,*
- *if $s\ h_S\ s'$ then the restriction $h_T\colon {}^\bullet s \to {}^\bullet s'$ and $h_T\colon s^\bullet \to s'^\bullet$ are total functions, and*
- *if $t' = h_T(t)$ then $h_S^{op}\colon {}^\bullet t' \to {}^\bullet t$ and $h_S^{op}\colon t'^\bullet \to t^\bullet$ are total functions, where h_S^{op} is the opposite relation to h_S.*

Morphisms among safe nets preserve the reachable markings: let $h\colon N \to N'$ be a net morphism. For each $m, m' \in \mathcal{M}_N$ and $A \in \mu T$, if $m\,[A\rangle\, m'$ then $h_S(m)\,[\mu h_T(A)\rangle\, h_S(m')$ where $h_S(m) = \{s' \in S' \mid \exists s \in m \text{ and } s\ h_S\ s'\}$.

The mapping which is the second component of the unfolding is a suitable morphism which is called folding:

Definition 16. *Let $N = \langle S, T, F, m\rangle$ and $N' = \langle S', T', F', m'\rangle$ be two nets and $h\colon N \to N'$ a net morphism. h is a* folding *iff h_T is total, h_S is a total function and for all $t \in T$, there are bijections between ${}^\bullet t$ and ${}^\bullet h_T(t)$, $t^\bullet$ and $h_T(t)^\bullet$ and between m and m'.*

Given a safe net $N = \langle S, T, F, m\rangle$, an unfolding is a pair (C, p) where $C = \langle B, E, F', m'\rangle$ is a causal net and p is a folding (see [3, 4]). Observe that if N is a multi-clock net (the partition mapping being $\nu(N)$), also C is a multi-clock net, the $\nu(C)$ being defined as follows: for $b \in B$. $\nu(C)(b) = \nu(N)(h_s(b))$.

The notion of folding we have defined above has to be specialised when flow unravel nets are considered. Control places are used to enforce dependencies among transitions, but they do not represent *resources*.

Definition 17. *Let $R = \langle S_i^R \cup S_c^R, T^R, F^R, m^R\rangle$ be a flow unravel net and $N = \langle S, T, F, m\rangle$ be multi clock net. Then $p\colon R \to N$ is a* flow-folding *morphism iff $p'\colon R|_{S_i^R} \to N$ is a folding morphism, where $p_T = p'_T$ and p'_S is the restriction of p_S to places in S_i^R, and for all $s_c \in S_c$ and for all $s \in S$. $\neg(s_c\ h_s\ s)$.*

A consequence of this definition is that in a flow-folding morphism control places are not associated to any place in the net onto which a flow unravel net is folded.

We are now ready to define a *flow* unfolding.

Definition 18. *Let $N = (\langle S, T, F, m\rangle, \nu)$ be a multi-clock net. Then a flow unfolding is the pair $(R = \langle S^r, T^r, F^r, m^r\rangle, p)$ where*

1. *$p\colon \langle S^r, T^r, F^r, m^r\rangle \to \langle S, T, F, m\rangle$ is a flow folding morphism,*
2. *$R = \langle S^r, T^r, F^r, m^r\rangle$ is a uniformly labelled flow unravel net with respect to $p_S|_{S_i^R}$,*
3. *$\forall\ t, t' \in T^R$. ${}^\bullet t = {}^\bullet t'$ and $p_T(t) = p_T(t') \Rightarrow t = t'$,*
4. *$\forall\ t, t' \in T^R$. ${}^\bullet t \cap S_i = {}^\bullet t' \cap S_i$ and $t^\bullet \cap S_i = t'^\bullet \cap S_i \Rightarrow t = t'$, and*
5. *$\forall\ t \in T^R$. $|{}^\bullet t \cap S_c| \leq |{}^\bullet t \cap S_i|$.*

Condition (3) is the usual on unfoldings: two transitions that have the same preset and are mapped onto the same transition of the original net are the same. Internal places are meant to represent the $i-th$ occurrence of a token in a given place of the unfolded net. In this view Condition (2) states that each internal place bears the *same* occurrence of tokens. Condition (4) enforces two transitions consuming and producing the same tokens to be the same transition and the last condition puts a bound on the number of control places. This condition will be more clear when we will effectively construct the unfolding of a multi-clock net.

The algorithm to construct this unfolding is similar to the one devised in the construction of a branching process [4]. However, in order to adapt it to our

purposes, we have to characterize the ingredients, namely the names of places (either control or internal ones), transitions, what a *co-set* of the net is and finally what the *possible extensions* are.

We start introducing the *neighborhood* of a transition.

Definition 19. *Let* $N = \langle S, T, F, m\rangle$ *be a net, and let* $t \in T$*, with* $\circlearrowleft(t) = \{t' \mid t' \in {}^\bullet s$ *or* $t' \in s'^\bullet$ *with* $s \in {}^\bullet t$ *and* $s' \in t^\bullet\} \cup \{t\}$ *we denote the* neighborhood *of* t*, namely the transitions* following *and* preceding t*, including* t*.*

The transitions in the neighborhood of t are used to find the *local name* of the occurrence of the transition in the unfolding, and the name is used to characterize also internal and control places. To this aim, we introduce an equivalence on words (on alphabets containing the names of transitions in the net we have to unfold). Let $N = (\langle S, T, F, m\rangle, \nu)$ be a multi-clock net, and let $T' \subseteq T$ be a subset of transitions, and let w, w' two words on $(T')^+$, the for all $t \in T'$, we say that $w \sim_t w'$ iff for all $s \in [\![{}^\bullet t]\!]$, $|proj(w, {}^\bullet s)| = |proj(w', {}^\bullet s)|$ and for all $s \in [\![t^\bullet]\!]$, $|proj(w, {}^\bullet s)| = |proj(w', {}^\bullet s)|$, and with $(\!|w|\!)_{\sim_t}$ we denote the equivalence class of the word w. Control places and transitions are pairs where the first component is an equivalence class of words on an alphabet of transitions (restricted to the transitions of an automata forming the multi-clock net) and the second component is a transition. The first component of a control place, the equivalance class, encodes all the equivalent (local) histories leading to the same future, represented by the name of the transition in the second component.

The internal places of this unfolding are easy to identify: consider a multi-clock net $N = (\langle S, T, F, m\rangle, \nu)$, then these places are of the following form: $\{(s, i) \mid s \in S$ and $i \in \mathbb{N}^+\}$, meaning that the place s got its i-th token.

We characterise now the possible extensions of a flow unravel net. Let $R = \langle S, T, F, m\rangle$ be a flow unravel net and let $X \in \mathcal{X}(R)$, then $[\![m_X]\!]$ is a *cut* of R. With respect to causal nets, where a cut is a maximal subset of pairwise concurrent places (conditions), here we have to consider the reached marking of the execution of the elements of a state (a configuration in the corresponding FES). A *co-set* A is a subset of a cut of R such that there exists a transition $t \in T$ such that $[\![{}^\bullet t]\!] = A$.

Definition 20. *Let* (R, p) *be a flow unfolding, with* $R = \langle S_i \cup S_c, T, F, m\rangle$*. The possible extensions of* (R, p) *are the transitions* $((\!|w|\!)_{\sim_t}, t)$*,* $w \in \circlearrowleft(t)^*$*, such that for all* $s \in [\![{}^\bullet t]\!]$

- *there exists* $((\!|\alpha_s|\!)_{\sim_t}, t) \in S_c$ *where* $\alpha_s = proj(w, \circlearrowleft(t) \cap T_{\nu^{-1}(\nu(s))})$
- $A = \{((\!|\alpha_s|\!)_{\sim_t}, t) \mid s \in {}^\bullet t\} \cup \{(s, |proj(w, {}^\bullet s)| + m(s)) \mid s \in {}^\bullet t\}$ *is a co-set of* R *such that* ${}^\bullet t = p_S(A)$*, and*
- $((\!|w|\!)_{\sim_t}, t)$ *does not already belong to* (R, p)*.*

Let us focus on the set of places $A = \{((\!|\alpha_s|\!)_{\sim_t}, t)\} \cup \{(s, |proj(w, {}^\bullet s)| + m(s))\}$, where clearly $\{((\!|\alpha_s|\!)_{\sim_t}, t)\} \subseteq S_c$ and $\{(s, |proj(w, {}^\bullet s)| + m(s))\} \subseteq S_i$. The internal places (those of the form (s, i)) are the instances of tokens consumed by the transition $((\!|w|\!)_{\sim_t}, t)$, whereas the $\{((\!|\alpha_s|\!)_{\sim_t}, t)\}$ are the control places used by this transition.

The algorithm to construct the flow unfolding is the one described in Table 1. In this algorithm, $NextPe(R,p)$ is used to find all the possible extensions according to Def. 20.

Table 1 The algorithm for constructing a flow unfolding

Input: A multi-clock net $N = (S, T, F, m)$
Output: The flow unfolding $\mathcal{FU} = (R, p)$, $R = \langle S_i \cup S_c, T', F', m\rangle$
begin:
$\mathcal{FU} \leftarrow (\langle\{(s,1) \mid m(s) = 1\}, \emptyset, \emptyset, \{(s,1) \mid m(s) = 1\}\rangle, p)$
where $p_S((s,1)) = s$
$pe \leftarrow NextPe(R,p)$
while $pe \neq \emptyset$ **do**
Add to $\mathcal{FU}$ a transition $((\!|w|\!)_{\sim_t}, t) \in pe$
and compute the places:
$S_i^{((\!|w|\!)_{\sim_t},t)} = \{(s,k) \mid k = |proj(w, {}^\bullet s)| + 1 + m(s)\}, \quad \forall s \in [\![t^\bullet]\!]$
$S_c^{((\!|w|\!)_{\sim_t},t)} = \{((\!|\alpha_s|\!)_{\sim_{t'}}, t') \mid \alpha_s \sim_{t'} proj(wt, \circlearrowleft(t) \cap T_{\overline{s}})\}, \forall s \in [\![t^\bullet]\!]$ and $\forall t' \in [\![s^\bullet]\!]$
and extend F' with:
$F'(x, ((\!|w|\!)_{\sim_t}, t)) = 1, \forall x \in A$ (see Def. 20) and 0 otherwise
$F'(((\!|w|\!)_{\sim_t}, t), s) = 1, \forall s \in (S_c^{((\!|w|\!)_{\sim_t},t)} \cup S_i^{((\!|w|\!)_{\sim_t},t)})$ and 0 otherwise
then update data structures
$pe \leftarrow (pe \setminus ((\!|w|\!)_{\sim_t}, t)) \cup NextPe(R,p)$
$S_i \leftarrow S_i \cup S_i^{((\!|w|\!)_{\sim_t},t)}$
$S_c \leftarrow S_c \cup S_c^{((\!|w|\!)_{\sim_t},t)}$
$T' \leftarrow T' \cup \{((\!|w|\!)_{\sim_t}, t)\}$
and extend p_T with $p_T(((\!|w|\!)_{\sim_t}, t)) = t$ and p_S with $p_S((s,k)) = s$ with $(s,k) \in S_i^{((\!|w|\!)_{\sim_t},t)}$ and undefined for the places in $S_c^{((\!|w|\!)_{\sim_t},t)}$
end while

The following proposition states that at each iteration the algorithm produces a flow unravel net, that means that no circularity is introduced among control places.

Proposition 7. *Let (R,p) be the result of the algorithm in Table 1. Then R is a flow unravel net.*

Proof. Algorithm initialization creates the net $R^0 = \langle S^0 = S_i^0 \cup S_c^0 = \{(s,1) \mid m(s) = 1\}, \emptyset, \emptyset, m = \{(s,1) \mid m(s) = 1\}\rangle$ and a set of possible extensions $pe = \{((\!|\epsilon|\!)_{\sim_t}, t)\}$, for all t enabled by the initial marking of N. R^0 is a trivial flow unravel net and equipped with p is a flow unfolding.

Assume that up to the n-th possible extension added we have a flow unravel net $R^n = \langle S_i^n \cup S_c^n, T^n, F^n, m = \{(s,1) \mid m(s) = 1\}\rangle$ and be $((\!|w|\!)_{\sim_t}, t)$ the $(n+1)$-th. We will show that adding $((\!|w|\!)_{\sim_t}, t)$ the properties of Def. 10 are fulfilled.

(1) and (2) follow from the definition of possible extensions, in particular, (2) holds because we always put a transition iff there is a co-set that can fire it and at

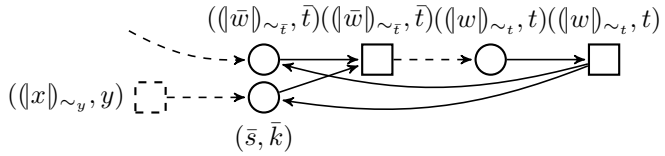
least one internal place is created in the process of adding a transition. In order to prove (3) we must ensure that if $(\!|\bar{w}|\!)_{\sim_{t'}}, t') \in T^m, m \leq n$ and $s_c \in (\!|\bar{w}|\!)_{\sim_{t'}}, t')^\bullet \cap {}^\bullet(\!|w|\!)_{\sim_t}, t) \cap S_c$ then there exists $s_i \in (\!|\bar{w}, t'|\!)_{\sim_{t'}})^\bullet \cap {}^\bullet(\!|w|\!)_{\sim_t}, t) \cap S_i$.

The possible extension $(\!|\bar{w}|\!)_{\sim_{t'}}, t')$ added a place (s, k') with $s \in t'^\bullet$, $k' = |proj(\bar{w}, {}^\bullet s)| + 1$, and the equivalences $\bar{w}t' \sim_t s_c \sim_t w$ exhibit the existence of $(s, k) \in A$ co-set of R^n, such that $s \in {}^\bullet t$ with $k = |proj(w, {}^\bullet s)|$. We show that $k = k'$ and thus Condition 3 of Def. 10 holds.

As $t' \in {}^\bullet s$, by using the definition of equivalence on places, we obtain:

$$\begin{aligned} \bar{w}t' \sim_t s_c \sim_t w &\Leftrightarrow |proj(\bar{w}t', {}^\bullet s)| = |proj(w, {}^\bullet s)| \\ &\Leftrightarrow |proj(\bar{w}, {}^\bullet s)| + 1 = |proj(w, {}^\bullet s)| \\ &\Leftrightarrow k' = k \end{aligned}$$

To prove Condition (5) of Def. 10 consider the following net fragment:



A cycle means that $\bar{w} \sim_{t'} wt$, where $(\!|wt|\!)_{\sim_{t'}}, t')$ is one of the control places created by adding the transition $(\!|w|\!)_{\sim_t}, t)$.

Before adding the possible extension $(\!|w|\!)_{\sim_t}, t)$ we can identify class representative of $u \sim_{\bar{t}}$ such that $|u|_t = n$ is the maximum number of t and $u \sim_{\bar{t}} \bar{w}$. The number of t is bounded because we assume that at least till $(\!|w|\!)_{\sim_t}, t)$ the net restricted to control places has no cycles. A state of $R^{n+1}|_{S_c^{n+1}}$ containing both $(\!|\bar{w}|\!)_{\sim_{\bar{t}}, t})$ and $(\!|w|\!)_{\sim_t}, t)$ would have the internal place $(\bar{s}, \bar{k})$ marked twice since its token is put by $(\!|x|\!)_{\sim_y, y})$ first, and then by $(\!|w|\!)_{\sim_t}, t)$. As for $(\!|\bar{w}|\!)_{\sim_{\bar{t}}, \bar{t}})$ we can find a class representative of $v \sim_{\bar{t}}$ such that $|v|_t$ is maximum and $v \sim_{\bar{t}} w$. Since the word v takes into account all the occurrences of t that u has, plus at least one, it is obvious that $\bar{k} = |proj(v, \{t, \dots\})| \neq |proj(u, \{t, \dots\})|$ and this leads to a contradiction. Hence a cycle cannot exist.

The algorithm does what it is expected: it gives a flow unfolding (it is routine to check the various conditions of Def. 18).

Theorem 1. *Let $N = (\langle S, T, F, m\rangle, \nu)$ be a multi-clock net. The algorithm in Table 1 constructs a flow unfolding for N.*

In Fig. 2 the result of few iterations of the algorithm applied to the multi-clock net in Fig. 1 is shown. Control places are depicted with a gray background. The transition $(\!|a|\!)_{\sim_b}, b)$ depends on the transition $(\!|\epsilon|\!)_{\sim_a}, a)$ as in the preset of the former there is the control place $(\!|a|\!)_{\sim_b}, b)$ that is in the postset of the latter. The transitions $(\!|a|\!)_{\sim_b}, b)$ and $(\!|\epsilon|\!)_{\sim_b}, b)$ are in conflict as they share, in their presets, the internal place $(p, 1)$, which is also a control place.

If we consider only internal places of a flow unfolding of a multi-clock net, we have again a multi-clock net.

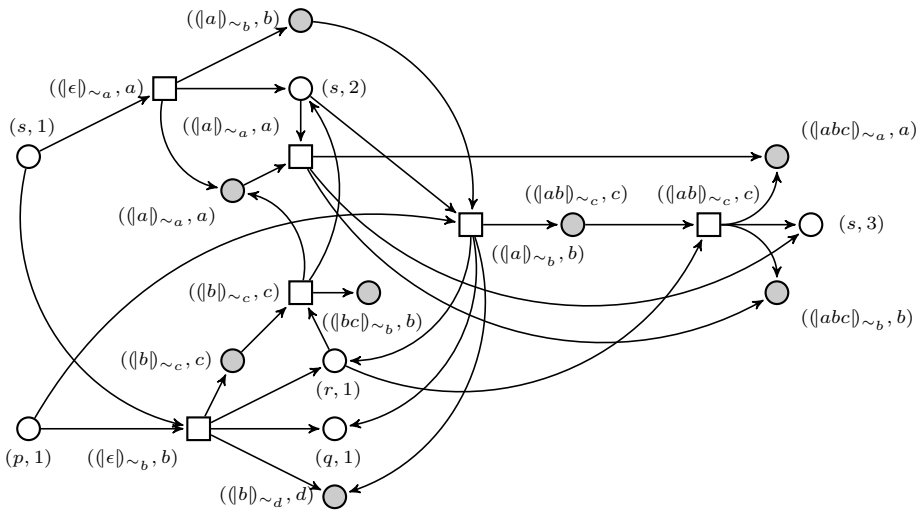


Fig. 2: A part of the flow unfolding of the multi-clock net shown in Fig. 1.

Proposition 8. *Let $N = (\langle S, T, F, m\rangle, \nu)$ be a multi-clock net and let (R, p) be the flow unfolding of N, where $R = \langle S_i^R \cup S_c^R, T^R, F^R, m^R\rangle$. Then $R|_{S_i^R}$ is a multi-clock net.*

Proof. The internal places of R are mapped onto the places of a multi-clock net. By definition of flow folding morphism it is clear that also $R|_{S_i^R}$ is a multi-clock net, defining $\nu(R|_{S_i^R})((s, k)) = p_S^{-1}(\nu(s))$.

Our construction has the same property of other notions of unfoldings: it is the most general one, in the sense that any other *candidate* to be an unfolding is uniquely mapped onto our flow unfolding.

To prove this we first have to define the *depth* of a flow unravel net $R = \langle S_i \cup S_c, T, F, m\rangle$. As the net when restricted to control places is acyclic it is straightforward to define the depth as follows:

Definition 21. *Let $R = \langle S_i \cup S_c, T, F, m\rangle$ be a flow unravel net. We define the* depth *of control places and transitions as follows:*

- *$depth(s) = 0$ if $s \in m$,*
- *$depth(t) = max\{depth(s) \mid s \in {}^\bullet t \cap S_c\} + 1$ and for all $s \in S_c$, and*
- *$depth(s) = min\{depth(t) \mid s \in t^\bullet\}$.*

There are two relevant differences with respect to the classical notion of depth: it is calculated on control places and not on internal places (because of cycle it may arise when considering internal places) and it is taken as the minimum among those available. Control places may have several incoming arcs modeling the various alternatives but locally equivalent past histories of a transition. It is then enough to consider the *shortest*.

Theorem 2. *Let N be a multi-clock net and (R, p) its flow unfolding. Let R' be a flow unravel net, $g : R' \to N$ be a morphism and let R' be uniformly labelled flow unravel net with respect to g_S. Then there exists a unique morphism from R' to R.*

Proof. We sketch the proof. The proof is done by constructing a chain of morphisms from suitable subnets of R' to R. Subnets are defined accordingly to the notion of prefix, which is in this case calculated as follows. Consider the control places at depth less or equal to n, and call them S'_n. Take $T'_n = {}^\bullet S'_n = \bigcup_{s \in S'_n} {}^\bullet s$, then the subnet of depth n of R' is $R'|_{T'_n}$.

Let us construct the morphisms as follows: $h^0 \colon R'|_{T'_0} \to R$ is defined as $h^0_T = \emptyset$ and the relation on places is $s\ h^0_S\ (g_S(s), 1)$, as T'_0 is empty (as g_S is a bijection on the initial marking $g_S(s)$ is well defined).

The morphism $h^{n+1} \colon R'|_{T'_{n+1}} \to R$ is obtained from h^n. Take any $t_1 \in T'_{n+1} \setminus T'_n$. We have two possibilities: either $g_T(t_1)$ is undefined or $g_T(t_1) = t$. In the first case $h^{n+1}_T(t_1)$ is undefined and the the places in $t_1{}^\bullet$ are unrelated to any place in R. In the second case, consider ${}^\bullet t_1$, and the places in $s' \in S^R_i \cup S^R_c$ such that $s\ h^n_S\ s'$, for $s \in {}^\bullet t_1$. By construction of the flow unfolding, these places are a co-set of R, and then there is a $t' \in T^R$ such that ${}^\bullet t'$ is this co-set and $p_T(t') = t = g_T(t_1)$. It is then easy to stipulate $h^{n+1}_T(t_1) = t$ and for all $s \in t_1{}^\bullet \cap S^R_i$ put $s h^{n+1}_T(g_S(s), k)$, where k is number associated to s in R' (as R' is uniformly labelled with respect to g_S). It remains to define the relation h^{n+1} on control places. Consider then any $t'_1 \in T'$ for which h^n_T is defined and such that $h^n_T(t'_1)^\bullet \cap {}^\bullet t' \cap S^R_c \neq \emptyset$. As N is multi-clock net $h^n_T(t'_1)^\bullet \cap {}^\bullet t' \cap S^R_c$ is a singleton, say $\{s_{h^n_T(t'_1), t'}\}$. Relate then the place in $t'_1{}^\bullet \cap {}^\bullet t_1 \cap S'_c$ with $s_{h^n_T(t'_1), t'}$. It is routine then to check that this is a well defined morphism.

Uniqueness can be proved along the same line.

5 Relating flow unfoldings to other unfoldings

In this section we relate our construction with two other notions of unfolding presented in literature, namely the one of *merged process* [7] and *trellises* [6]. Clearly a classical unfolding can be *folded* onto a flow unfolding by transforming the causal net into a flow unravel net and using Th. 2.

Let $N = \langle S, T, F, m \rangle$ be a safe net, (C, p) be an unfolding of N where $C = \langle B, E, F', m' \rangle$ is a causal net and $p \colon C \to N$ is a folding morphism (*i.e.* $\forall\ e, e' \in E.\ ({}^\bullet e = {}^\bullet e'$ *and* $p_T(e) = p_T(e')) \ \Rightarrow\ e = e'$), then (C, p) is called a *branching process* of N [4]. As p_S is a total function from B to S, it can be seen as a labelling function, hence we can associate to each place a number that it is called the *occurrence-depth* and that it is defined as follows: the occurrence-depth of a condition $b \in B$ is the highest number of equally labelled conditions that are on a path from an initial condition to b. We recall the definition of merged process for safe nets:

Definition 22. *Let N be a safe net and (C, p) be branching process. The merged process of (C, p) is the net $\mathfrak{Merge}(C, p)$ defined by the following steps:*

1. *all the conditions bearing the same label and having the same* occurrence-depth *are* fused *together, and these conditions, called* mp-*conditions, inherits the same incoming and outgoing arcs of the conditions that are fused, finally an* mp-*condition inherit the same label of the fused conditions,*
2. *after performing the previous step, all the transitions with the same label, the same preset and the same postset are* fused *together, giving an* mp-*event, and they inherit the label from the fused as well as the incoming and outgoing arcs, and*
3. *the initial marking is given by the* mp-*conditions which are originated by conditions that were minimal in the causal net C.*

Interestingly enough, our construction is strongly related to this one:

Theorem 3. *Let N be a multi-clock net, (R, p) its flow unfolding, (C, f) be its branching process and $\mathfrak{Merge}(C, f)$ the associated merged process. Then $R|_{S_i}$ and $\mathfrak{Merge}(C, f)$ are the same net up to renaming of places and transitions.*

Proof. We give the proof idea. Clearly internal places of R are the same of $\mathfrak{Merge}(C, f)$, being N a multi-clock net (hence a place belongs to a unique automaton, as automata synchronize on common transitions). We have to show that the transitions are the same as well. Assume they are not, then R has at least two transitions t, t' which are identified in $\mathfrak{Merge}(C, f)$ but are not in R. As they are identified in $\mathfrak{Merge}(C, f)$ they the same *internal* preset and postset. But this is impossible. Assume now that there are two transitions t, t' in $\mathfrak{Merge}(C, f)$ which are identified in R. Then they have the same internal preset and postset hence they have to be identified also in $\mathfrak{Merge}(C, f)$. Hence the thesis.

A merged process of a multi-clock net can be transformed into a flow unfolding as well, but control places have to be added with a different criteria with respect to the one devised for causal net (*i.e.* by looking directly if two transitions have a place in common). The idea is that there is a control place among two transitions if they share a place and they belong to the same configuration. Consider the merged process in Fig. 3(d). The multiset of transition X defined as $X(a_1) = 1$, $X(c_1) = 2$ and $X(b_2) = 2$ is a state of the net depicted in Fig. 3(d), but it is not a configuration of the causal net in Fig. 3(c). In [7] a way to check if an *run* of the net is a configuration is provided without resorting to the branching process, that can be used to add the control places.

A different consideration have to be done for trellises. Trellises are defined for multi-clock nets, and they unfold properly the time but do not expand conflicts. In a trellis each execution corresponds to the synchronization of trajectories in each component, synchronization performed on appropriate equally labelled transitions. Now conditions in a trellis are fused only if they bear the same label (are occurrence of the same place) and have the same *time* and this time is calculated counting the conditions belonging to the same automata in the past. Thus they are not easily comparable with flow unfoldings, where time does not count.

In Fig. 3 we draw (parts) of the unfolding/branching process (c), merged process (d), trellis (e) and flow unfolding (b) of the net (a). Internal places and

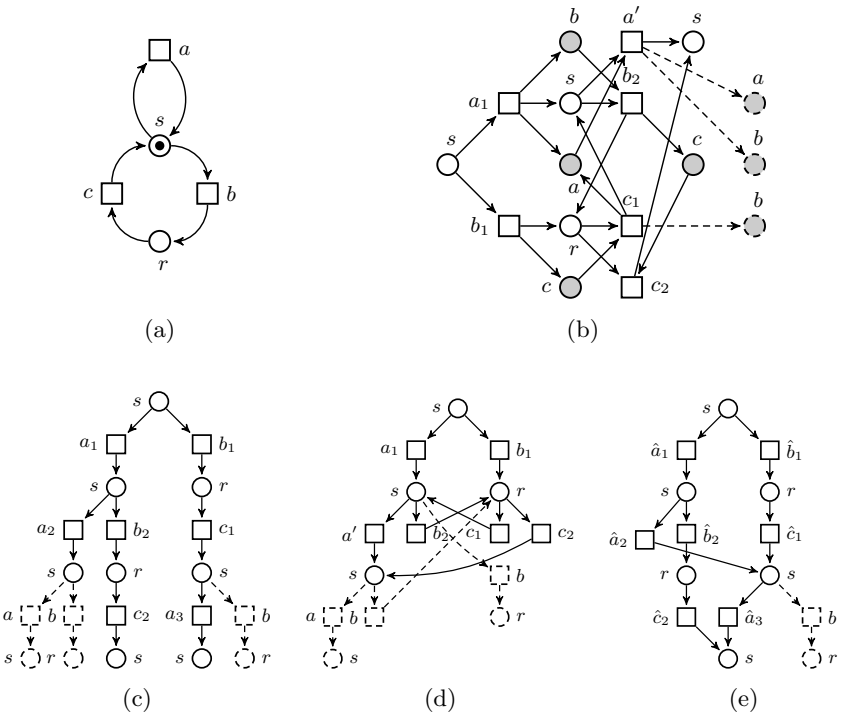


Fig. 3: A multi-clock net and various unfoldings.

transitions of (b) give the net drawn in (d). In these unfoldings there is no concurrency, only conflicts and causal dependencies may be deduced. On the trellis the dependencies are deduced by going from the initial condition along a path, similarly to the flow unfolding, and conflicts arise from the use of a common resource (hence in the trellis (e) the events $\hat{a}_1$ and $\hat{b}_1$ are in conflict). In the merged process executable cycles are possible: b_2 can be executed twice (one after a_1 and the second after c_1), whereas this is impossible in the flow unravel net. The event a' is obtained fusing a_2 and a_3 in the branching process. In the flow unfolding b_2 can be executed only once, and again, like in trellises, conflicts arise by using the same resource: b_2 is in conflict with a' (which is again the result of fusing a_2 and a_3) but now b_2 cannot be executed twice, as it uses the token that the transition a_1 puts in the control place b. Dependencies are captured along the control places that in this figure are labelled with the name of the transitions they enable.

6 Conclusions

In this paper we have proposed a notion of unfolding of multi-clock nets that can be easily related to the class of flow event structures. This unfolding is

proved to be the more general one among those for which a dependency and a conflict relation can be defined (see Th. 2), and this result gives, in our opinion, evidence that the construction is reasonable and useful. Indeed our result gives further substance from a theoretical viewpoint to merged processes (see Th. 3). Moreover the relation of causal dependency may be relevant for model checking performed using this kind of unfolding.

Still our construction has some weakeness. It relies on multi-clock nets, and though any safe net can be turned into a multi-clock net, as already noticed in the introduction, the construction as it is now cannot be applied to unsafe nets in general. The reason is that control places are used to equate histories leading to the same future, and these histories are easily identified just looking at transitions in a component. This information turn out to be rather crucial. The generalization we devise should consider an adequate equivalence to equate *local* histories. In this line, the decomposition approach proposed in [23] has to be pursued. Furthermore, the number of control places depends on the number of components of a multi-clock net, and the construction that associates a multi-clock net to a safe one creates a *component* for each place in the original net.

We have certainly left out the investigation on how to find *cut-off* events to obtain a *finite* and *complete* prefix of a flow unfolding, which is already presented in [22] for the merged processes. The clear dependency relation we propose could fit in the theoretical framework devised in [24].

Another advantage of having a dependency relation is the possibility of applying the approach pursued in [25] and [26]. We are confident that reveal relations can be defined starting from a flow unravel nets, and this will be the subject of future works.

Acknowledgement: We would like to thank Eric Fabre and Victor Khomenko for many useful discussions on the topic of this paper, and to the reviewers as well for the useful criticisms that have helped us in improving the paper.

References

1. Khomenko, V.: Model Checking based on Prefixes of Petri Net Unfoldings. PhD thesis, School of Computing Science, University of Newcastle upon Tyne (2003)
2. Fabre, E., Benveniste, A., Haar, S., Jard, C.: Distributed Monitoring of Concurrent and Asynchronous Systems. Discrete Event Dynamic Systems **15** (2005) 33–84
3. Winskel, G.: Event Structures. In Brauer, W., Reisig, W., Rozenberg, G., eds.: Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, LNCS 255, Springer Verlag (1987) 325–392
4. Engelfriet, J.: Branching processes of Petri nets. Acta Informatica **28** (1991) 575–591
5. McMillan, K.: A Technique of State Space Search Based on Unfolding. Formal Methods in System Design **6** (1995) 45–65
6. Fabre, E.: Trellis processes : A compact representation for runs of concurrent systems. Discrete Event Dynamic Systems **17** (2007) 267–306
7. Khomenko, V., Kondratyev, A., Koutny, M., Vogler, W.: Merged Processes: a new condensed representation of Petri net behaviour. Acta Informatica **43** (2006) 307–330

8. van Glabbeek, R.J., Plotkin, G.D.: Configuration structures, event structures and Petri nets. Theoretical Computer Science **410** (2009) 4111–4159
9. Pinna, G.M., Poigné, A.: On the nature of events: another perspective in concurrency. Theoretical Computer Science **138** (1995) 425–454
10. Boudol, G.: Flow Event Structures and Flow Nets. In Guessarian, I., ed.: Semantics of Systems of Concurrent Processes. LNCS 469, Springer (1990) 62–95
11. Boudol, G., Castellani, I.: Flow models of distributed computations: Three equivalent semantics for ccs. Information and Computation **114** (1994) 247–314
12. Baldan, P., Corradini, A., Montanari, U.: Contextual Petri nets, asymmetric event structures and processes. Information and Computation **171** (2001) 1–49
13. Baldan, P., Busi, N., Corradini, A., Pinna, G.M.: Domain and event structure semantics for Petri nets with read and inhibitor arcs. Theoretical Computer Science **323** (2004) 129–189
14. Langerak, R.: Bundle Event Structures: A Non-Interleaving Semantics for Lotos. In Diaz, M., Groz, R., eds.: Fifth International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, FORTE '92, IFIP Transactions C-10, North-Holland (1992) 331–346
15. Gunawardena, J.: A generalized event structure for the Muller unfolding of a safe net. In Best, E., ed.: CONCUR'93 Conference Proceedings. LNCS 715, Springer Verlag (1993) 278–292
16. Pinna, G.M.: How much is worth to remember? a taxonomy based on Petri Nets Unfoldings. In Kristensen, L.M., Petrucci, L., eds.: Proceedings of the 32nd International Conference on Application and Theory of Petri Nets. LNCS 6709, Springer Verlag (2011) 109–128
17. van Glabbeek, R.J., Plotkin, G.D.: Configuration structures. In Kozen, D., ed.: Proceedings of 10^{th} Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press (1995) 199–209
18. Reisig, W.: Petri Nets: An Introduction. EACTS Monographs on Theoretical Computer Science. Springer Verlag (1985)
19. Hayman, J., Winskel, G.: The unfolding of general Petri nets. In Hariharan, R., Mukund, M., Vinay, V., eds.: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008), Dagstuhl, Germany, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2008)
20. Nielsen, M., Plotkin, G., Winskel, G.: Petri Nets, Event Structures and Domains, Part 1. Theoretical Computer Science **13** (1981) 85–108
21. Esparza, J., Römer, S., Vogler, W.: An Improvement of McMillan's Unfolding Algorithm. Formal Methods in System Design **20** (2002) 285–310
22. Khomenko, V., Mokhov, A.: Direct construction of Complete Merged Processes. The Computer Journal (2013) to appear
23. Rathke, J., Sobocinski, P., Stephens, O.: Decomposing Petri nets. CoRR **abs/1304.3121** (2013)
24. Khomenko, V., Koutny, M., Vogler, W.: Canonical prefixes of Petri net unfoldings. Acta Informatica **40** (2003) 95–118
25. Balaguer, S., Chatain, T., Haar, S.: Building occurrence nets from reveals relations. Fundamamenta Informaticae **123** (2013) 245–272
26. Haar, S., Kern, C., Schwoon, S.: Computing the reveals relation in occurrence nets. Theoretical Computer Science **493** (2013) 66–79