Concurrency and Probability: Removing Confusion, Compositionally

Roberto Bruni
Dipartimento di Informatica
Università di Pisa
Pisa, Italy
bruni@di.unipi.it

Hernán Melgratti
Departamento de Computación
Universidad de Buenos Aires & Conicet
Buenos Aires, Argentina
hmelgra@dc.uba.ar

Ugo Montanari
Dipartimento di Informatica
Università di Pisa
Pisa, Italy
ugo@di.unipi.it

Abstract

Assigning a satisfactory truly concurrent semantics to Petri nets with confusion and distributed decisions is a long standing problem, especially if one wants to resolve decisions by drawing from some probability distribution. Here we propose a general solution based on a recursive, static decomposition of (occurrence) nets in loci of decision, called *structural branching cells* (*s-cells*). Each s-cell exposes a set of alternatives, called *transactions*. Our solution transforms a given Petri net into another net whose transitions are the transactions of the s-cells and whose places are those of the original net, with some auxiliary structure for bookkeeping. The resulting net is confusion-free, and thus conflicting alternatives can be equipped with probabilistic choices, while nonintersecting alternatives are purely concurrent and their probability distributions are independent. The validity of the construction is witnessed by a tight correspondence with the recursively stopped configurations of Abbes and Benveniste. Some advantages of our approach are that: i) s-cells are defined statically and locally in a compositional way; ii) our resulting nets faithfully account for concurrency.

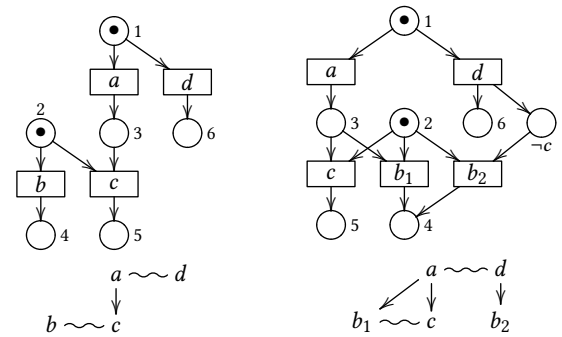
CCS Concepts • **Theory of computation → Parallel computing models**; *Probabilistic computation*;

Keywords Petri nets, confusion, dynamic nets, persistent places, OR causality

ACM Reference Format:
Roberto Bruni, Hernán Melgratti, and Ugo Montanari. 2018. Concurrency and Probability: Removing Confusion, Compositionally. In *LICS '18: 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, July 9–12, 2018, Oxford, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209108.3209202

1 Introduction

Concurrency theory and practice provide a useful abstraction for the design and use of a variety of systems. Concurrent computations (also *processes*), as defined in many models, are equivalence classes of execution sequences, called *traces*, where the order of concurrent (i.e., independent) events is inessential. A key notion in concurrent models is *conflict* (also known as choices or decisions). Basically, two events are in conflict when they cannot occur in the same execution. The interplay between concurrency and conflicts introduces a phenomenon in which the execution of an event can be influenced by the occurrence of another concurrent (and hence independent) event. Such situation, known as *confusion*, naturally arises in concurrent and distributed systems and is intrinsic to problems involving mutual exclusion [Smith 1996]. When interleaving semantics is considered, the problem is less compelling, however it has been recognised and studied from the beginning of net research [Rozenberg and Engelfriet 1998], and to address it in a general and acceptable way can be considered as a long-standing open problem for concurrency theory.



(a) Asymmetric confusion **(b)** Removing confusion

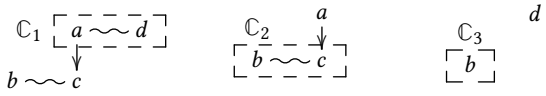
Figure 1. Some nets (top) and their event structures (bottom)

To illustrate confusion, we rely on Petri nets [Petri 1962; Reisig 2013], which are a basic, well understood model of concurrency. The simplest example of (asymmetric) confusion is the net in Fig. 1a. The net has two traces involving the concurrent events a and b, namely $\sigma_1 = a;b$ and $\sigma_2 = b;a$. Both traces define the same concurrent execution. Contrastingly, σ_1 and σ_2 are associated with completely different behaviours of the system as far as the resolution of choices is concerned. In fact, the system makes two choices while executing σ_1: firstly, it chooses a over d, which enables c; secondly, b is selected over c. Differently, the system makes just one choice in σ_2: since c is not enabled, b is executed without any choice; after that, the system chooses a over d. As illustrated by this example, the choices made by two different traces of the same concurrent computation may differ depending on the order in which concurrent events occur.

The fundamental problem behind confusion relates to the description of distributed, global choices. Such problem becomes essential when choices are driven by probabilistic distributions and one wants to assign probabilities to executions, as it is the case with probabilistic, concurrent models. Consider again Fig. 1a and assume that a is chosen over d with probability p_a while b is chosen over c with probability p_b When treated as independent choices, the

LICS '18, July 9–12, 2018, Oxford, United Kingdom

ACM ISBN 978-1-4503-5583-4/18/07…$15.00
https://doi.org/10.1145/3209108.3209202



(a) Initial configuration **(b)** a is chosen **(c)** d is chosen

Figure 2. AB's dynamic branching cells for the example in Fig. 1a

trace σ_1 has probability $p_a \cdot p_b$, while σ_2 has probability $1 \cdot p_a = p_a$. Hence, two traces of the same concurrent computation, which are deemed equivalent, would be assigned different probabilities.

Different solutions have been proposed in the literature for adding probabilities to Petri nets [Bouillard et al. 2009; Dugan et al. 1984; Eisentraut et al. 2013; Haar 2002; Katoen et al. 1993; Kudlek 2005; Marsan et al. 1984; Molloy 1985]. To avoid confusion, most of them replace nondeterminism with probability only in part, or disregard concurrency, or introduce time dependent stochastic distributions, thus giving up the time and speed independence features typical of truly concurrent models. Confusion-free probabilistic models have been studied in [Varacca et al. 2006], but this class, which subsumes free-choice nets, is usually considered quite restrictive. More generally, the distributability of decisions has been studied, e.g., in [Katoen and Peled 2013; van Glabbeek et al. 2013], but while the results in [van Glabbeek et al. 2013] apply to some restricted classes of nets, the approach in [Katoen and Peled 2013] requires nets to be decorated with agents and produces distributed models with both nondeterminism and probability, where concurrency depends on the scheduling of agents.

A substantial advance has been contributed by Abbes and Benveniste (AB) [Abbes and Benveniste 2005, 2006, 2008]. They consider prime event structures and provide a *branching cell decomposition* that establishes the order in which choices are resolved (see Section 4.2). Intuitively, the event structure in Fig. 1a has the three branching cells outlined in Fig. 2. First a decision between a and d must be taken (Fig. 2a): if a is executed, then a subsequent branching cell $\{b, c\}$ is enabled (Fig. 2b); otherwise (i.e., if d is chosen) the branching cell $\{b\}$ is enabled (Fig. 2c). In this approach, the trace $\sigma_2 = b; a$ is not admissible, because the branching cell $\{b\}$ does not exist in the original decomposition (Fig. 2a): it appears *after* the choice of d over a. Branching cells are equipped with independent probability distributions and the probability assigned to a concurrent execution is given by the product of the probabilities assigned by its branching cells. Notably, the sum of the probabilities of maximal configurations is 1. Every decomposition of a configuration yields an execution sequence compatible with that configuration. Unfortunately, certain sequences of events, legal w.r.t. the configuration, are not executable according to AB.

Problem statement The question addressed in this paper is a foundational one: *can concurrency and general probabilistic distributions coexist in Petri nets? If so, under which circumstances?* By *coexistence* we mean that all the following issues must be addressed:

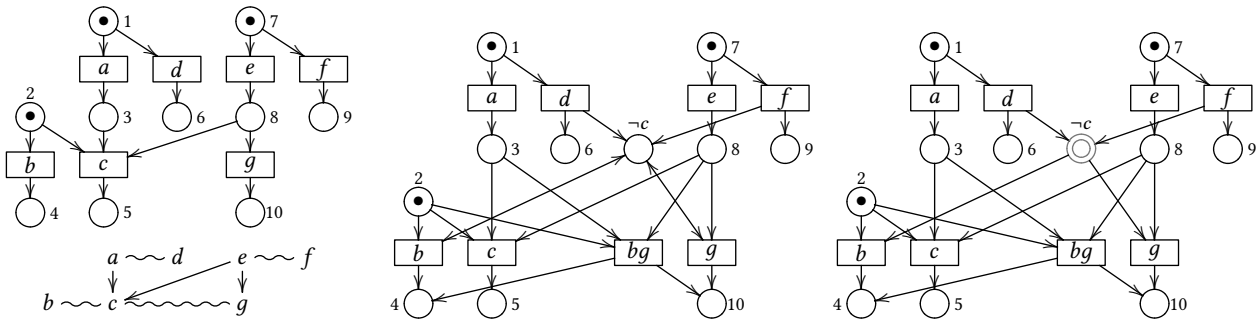
1. *Speed independence*: Truly concurrent semantics usually requires computation to be time-independent and also independent from the relative speed of processes. In this sense, while attaching rates of stochastic distributions to transitions is perfectly fine with interleaving semantics, they are not suited when truly concurrent semantics is considered.
2. *Schedule independence*: Concurrent events must be driven by independent probability distributions.
3. *Probabilistic computation*: Nondeterministic choices must be replaceable by probabilistic choices. This means that whenever two transitions are enabled, the choice to fire one instead of the other is either inessential (because they are concurrent) or is driven by a probability distribution.
4. *Complete concurrency*: It must be possible to establish a bijective correspondence between equivalence classes of firing sequences and a suitable set of concurrent processes.
5. *Sanity check #1*: All firing sequences of the same process carry the same probability, i.e., the probability of a concurrent computation is independent from the order of execution.
6. *Sanity check #2*: The sum of the probabilities assigned to all possible processes must be 1.

In this paper we provide a positive answer for finite occurrence nets: given any such net we show how to define loci of decisions, called *structural branching cells* (*s-cells*), and construct another net where independent probability distributions can be assigned to concurrent events. This means that each s-cell can be assigned to a distributed random agent and that any concurrent computation is independent from the scheduling of agents.

Overview of the approach Following the rationale behind AB's approach, a net is transformed into another one that postpones the execution of choices that can be affected by pending decisions. According to this intuition, the net in Fig. 1a is transformed into another one that delays the execution of b until all its potential alternatives (i.e., c) are enabled or definitively excluded. In this sense, b should never be executed before the decision between a and d is taken, because c could still be enabled (if a is chosen). As a practical situation, imagine that a and d are the choices of your partner to either come to town (a) or go to the sea (d) and that you can go to the theatre alone (b), which is always an option, or go together with him/her (c), which is possible only when he/she is in town and accepts the invitation. Of course you better postpone the decision until you know if your partner is in town or not. This behaviour is faithfully represented, e.g., by the confusion-free net in Fig. 1b, where two variants of b are made explicit: b_1 (your partner is in town) and b_2 (your partner is not in town). The new place $\neg c$ represents the fact that c will never be enabled. Now, from the concurrency point of view, there is a single process that comprises both a and b_1 (with a a cause of b_1), whose overall probability is the product of the probability of choosing a over d by the probability of choosing b_1 over c. The other two processes comprise, respectively, d and b_2 (with d a cause of b_2) and a and c (with a a cause of c). As the net is confusion-free all criteria in the desiderata are met.

The general situation is more involved because: i) there can be several ways to disable the same transition; ii) resolving a choice may require to execute several transitions at once. Consider the net in Fig. 3a: i) c is discarded as soon as d *or* f fires; and ii) when both a and e are fired we can choose to execute c alone or *both* b and g. Likewise the previous example, we may expect to transform the net as in Fig. 3b. Again, the place $\neg c$ represents the permanent disabling of c. This way a probability distribution can drive the choice between c and (the joint execution of) bg, whereas b and g (if enabled) can fire concurrently when $\neg c$ is marked.

A few things are worth remarking: i) a token in $\neg c$ can be needed several times (e.g., to fire b and g), hence tokens should be read but



(a) Confusion with OR-causes **(b)** An attempt **(c)** A solution with persistent places

Figure 3. Running example

not consumed from $\neg c$ (whence the double headed arcs from $\neg c$ to b and g, called *self-loops*); ii) several tokens can appear in the place $\neg c$ (by firing both d and f). These facts have severe repercussions on the concurrent semantics of the net. Suppose the trace $d; f; b$ is observed. Does b causally depend on the token generated from d or from f (or from both)? Moreover, consider the trace $d; e; b; g$, in which b takes and releases a token in $\neg c$. Does g causally depend on b (due to such self-loop)? This last question can be solved by replacing self-loops with *read arcs* [Montanari and Rossi 1995], so that the firing of b does not alter the content of $\neg c$ and thus no causal dependency arises between b and g. Nevertheless, if process semantics or event semantics is considered, then we should explode all possible combinations of causal dependencies, thus introducing a new, undesired kind of nondeterminism. In reality, we should not expect any causal dependency between b and g, while both have OR dependencies on d and f.

To account for OR dependencies, we exploit the notion of *persistence*: tokens in a persistent place have infinite weight and are collective. Namely, once a token reaches a persistent place, it cannot be removed and if two tokens reach the same persistent place they are indistinguishable. Such networks are a variant of ordinary P/T nets and have been studied in [Crazzolara and Winskel 2005]. In the example, we can declare $\neg c$ to be a persistent place and replace self-loops/read arcs on $\neg c$ with ordinary outgoing arcs (see Fig. 3c). Nicely we are able to introduce a process semantics for nets with persistent places that satisfies complete concurrency.

The place $\neg c$ in the examples above is just used to sketch the general idea: our transformation introduces persistent places like $\overline{3}$ to express that a token will never appear in the regular place 3.

Contribution. In this paper we show how to systematically derive confusion-free nets (with persistency) from any (finite, occurrence) Petri net and equip them with probabilistic distributions and concurrent semantics in the vein of AB's construction.

Technically, our approach is based on a structurally recursive decomposition of the original net in s-cells. A simple kind of Asperti-Busi's dynamic nets is used as an intermediate model to structure the coding. While not strictly necessary, the intermediate step emphasises the hierarchical nature of the construction. The second part is a general flattening step independent of our special case. Our definition is purely local (to s-cells), static and compositional, whereas AB's is dynamic and global (i.e., it requires the entire PES). Using nets with persistency, we compile the execution strategy of nets with confusion in a statically defined, confusion-free, operational model. The advantage is that the concurrency within a process of the obtained p-net is consistent with execution, i.e., all linearizations of a persistent process are executable.

Structure of the paper After fixing notation in Section 2, our solution to the confusion problem consists of the following steps: (i) we define s-cells in a compositional way (Section 3.1); (ii) from s-cells decomposition and the use of dynamic nets, we derive a confusion-free net with persistency (Section 3.2); (iii) we prove the correspondence with AB's approach (Section 4); (iv) we define a new notion of process that accounts for OR causal dependencies and satisfies complete concurrency (Section 5); and (v) we show how to assign probabilistic distributions to s-cells (Section 6).

2 Preliminaries

2.1 Notation

We let $\mathbb{N}$ be the set of natural numbers, $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$ and $\mathbb{2} = \{0, 1\}$. We write U^S for the set of functions from S to U: hence a subset of S is an element of $\mathbb{2}^S$, a multiset m over S is an element of $\mathbb{N}^S$, and a bag b over S is an element of $\mathbb{N}_\infty^S$. By overloading the notation, union, difference and inclusion of sets, multisets and bags are all denoted by the same symbols: $\cup$, $\setminus$ and $\subseteq$, respectively. In the case of bags, the difference $b \setminus m$ is defined only when the second argument is a multiset, with the convention that $(b \setminus m)(s) = \infty$ if $b(s) = \infty$. Similarly, $(b \cup b')(s) = \infty$ if $b(s) = \infty$ or $b'(s) = \infty$. A set can be seen as a multiset or a bag whose elements have unary multiplicity. Membership is denoted by $\in$: for a multiset m (or a bag b), we write $s \in m$ for $m(s) \neq 0$ ($b(s) \neq 0$). Given a relation $R \subseteq S \times S$, we let R^+ be its transitive closure and R^* be its reflexive and transitive closure. We say that R is *acyclic* if $\forall s \in S.\ (s, s) \notin R^+$.

2.2 Petri Nets, confusion and free-choiceness

A *net structure* N (also *Petri net*) [Petri 1962; Reisig 2013] is a tuple (P, T, F) where: P is the set of places, T is the set of transitions, and $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation. For $x \in P \cup T$, we denote by ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{z \mid (x, z) \in F\}$ its *pre-set* and *post-set*, respectively. We assume that P and T are disjoint and non-empty and that ${}^\bullet t$ and $t^\bullet$ are non empty for every $t \in T$. We write $t : X \to Y$ for $t \in T$ with $X = {}^\bullet t$ and $Y = t^\bullet$.

A *marking* is a multiset $m \in \mathbb{N}^P$. We say that p is *marked* at m if $p \in m$. We write (N, m) for the net N *marked* by m. We write m_0 for the initial marking of the net, if any.

Graphically, a Petri net is a directed graph whose nodes are the places and transitions and whose set of arcs is F. Places are drawn as circles and transitions as rectangles. The marking m is represented by inserting $m(p)$ tokens in each place $p \in m$ (see Fig. 1).

A transition t is *enabled* at the marking m, written $m \xrightarrow{t}$, if ${}^\bullet t \subseteq m$. The execution of a transition t enabled at m, called *firing*, is written $m \xrightarrow{t} m'$ with $m' = (m \setminus {}^\bullet t) \cup t^\bullet$. A firing sequence from m to m' is a finite sequence of firings $m = m_0 \xrightarrow{t_1} \cdots \xrightarrow{t_n} m_n = m'$, abbreviated to $m \xrightarrow{t_1 \cdots t_n} m'$ or just $m \to^* m'$. Moreover, it is *maximal* if no transition is enabled at m'. We write $m \xrightarrow{t_1 \cdots t_n}$ if there is m' such that $m \xrightarrow{t_1 \cdots t_n} m'$. We say that m' is *reachable* from m if $m \to^* m'$. The set of markings reachable from m is written $[m\rangle$. A marked net (N, m) is *safe* if each $m' \in [m\rangle$ is a set.

Two transition t, u are in *direct conflict* if ${}^\bullet t \cap {}^\bullet u \neq \emptyset$. A net is called *free-choice* if for all transitions t, u we have either ${}^\bullet t = {}^\bullet u$ or ${}^\bullet t \cap {}^\bullet u = \emptyset$, i.e., if a transition t is enabled then all its conflicting alternatives are also enabled. Note that free-choiceness is purely structural. Confusion-freeness considers instead the dynamics of the net. A safe marked net (N, m_0) has *confusion* iff there exists a reachable marking m and transitions t, u, v such that:

1. (i) t, u, v are enabled at m, (ii) ${}^\bullet t \cap {}^\bullet u \neq \emptyset \neq {}^\bullet u \cap {}^\bullet v$, (iii) ${}^\bullet t \cap {}^\bullet v = \emptyset$ (*symmetric case*); or
2. (i) t and v are enabled at m, (ii) u is not enabled at m but it becomes enabled after the firing of t, and (iii) ${}^\bullet t \cap {}^\bullet v = \emptyset$ and ${}^\bullet v \cap {}^\bullet u \neq \emptyset$ (*asymmetric case*).

In case 1, t and v are concurrently enabled but the firing of one disables an alternative (u) to the other. In case 2, the firing of t enables an alternative to u. An example of symmetric confusion is given by $m = \{2, 3, 8\}$, $t = b$, $u = c$ and $v = g$ in Fig. 3a, while for the asymmetric case take $m = \{1, 2\}$, $t = a$, $v = b$ and $u = c$ in Fig. 1a. A net is *confusion-free* when it has no confusion.

2.3 Deterministic Nonsequential Processes

A *deterministic nonsequential process* (or just *process*) [Goltz and Reisig 1983] represents the equivalence class of all firing sequences of a net that only differ in the order in which concurrent firings are executed. It is given as a mapping $\pi : \mathcal{D} \to N$ from a *deterministic occurrence net* $\mathcal{D}$ to N (preserving pre- and post-sets), where a deterministic occurrence net is such that: (1) the flow relation is acyclic, (2) there are no backward conflicts ($\forall p \in P.\ |{}^\bullet p| \leq 1$), and (3) there are no forward conflicts ($\forall p \in P.\ |p^\bullet| \leq 1$). We let ${}^\circ\mathcal{D} = \{p \mid {}^\bullet p = \emptyset\}$ and $\mathcal{D}^\circ = \{p \mid p^\bullet = \emptyset\}$ be the sets of *initial* and *final places* of $\mathcal{D}$, respectively (with $\pi({}^\circ\mathcal{D})$ be the initial marking of N). When N is an acyclic safe net, the mapping $\pi : \mathcal{D} \to N$ is just an injective graph homomorphism: without loss of generality, we name the nodes in $\mathcal{D}$ as their images in N and let π be the identity. The firing sequences of a processes $\mathcal{D}$ are its maximal firing sequences starting from the marking ${}^\circ\mathcal{D}$. A process of N is *maximal* if its firing sequences are maximal in N.

For example, take the net in Fig. 1a. The equivalence class of the firing sequences $m_0 \xrightarrow{a\,b}$ and $m_0 \xrightarrow{b\,a}$ is the maximal process $\mathcal{D}$ with places $\{1, 2, 3, 4\}$ and transitions $\{a : 1 \to 3, b : 2 \to 4\}$, where ${}^\circ\mathcal{D} = \{1, 2\}$ and $\mathcal{D}^\circ = \{3, 4\}$.

Given an acyclic net we let $\preceq = F^*$ be the (reflexive) *causality* relation and say that two transitions t_1 and t_2 are in *immediate conflict*, written $t_1 \#_0 t_2$ if $t_1 \neq t_2 \ \wedge \ {}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$. The *conflict relation* $\#$ is defined by letting $x \# y$ if there are $t_1, t_2 \in T$ such that $(t_1, x), (t_2, y) \in F^+$ and $t_1 \#_0 t_2$. Then, a *nondeterministic occurrence net* (or just *occurrence net*) is a net $O = (P, T, F)$ such that: (1) the flow relation is acyclic, (2) there are no backward conflicts ($\forall p \in P.\ |{}^\bullet p| \leq 1$), and (3) there are no self-conflicts ($\forall t \in T.\ \neg(t \# t)$). The *unfolding* $\mathcal{U}(N)$ of a safe Petri net N is an occurrence net that accounts for all (finite and infinite) runs of N: its transitions model all the possible instances of transitions in N and its places model all the tokens that can be created in any run. Our construction takes a finite occurrence net as input, which can be, e.g., the (truncated) unfolding of any safe net.

2.4 Nets With Persistency

Nets with persistency (*p-nets*) [Crazzolara and Winskel 2005] partition the set of places into regular places P (ranged by $p, q, \ldots$) and persistent places $\mathbf{P}$ (ranged by $\mathbf{p}, \mathbf{q}, \ldots$). We use s to range over $\mathbb{S} = P \cup \mathbf{P}$ and write a p-net as a tuple $(\mathbb{S}, T, F)$. Intuitively, persistent places guarantee some sort of monotonicity about the knowledge of the system. Technically, this is realised by letting states be bags of places $b \in \mathbb{N}_\infty^{\mathbb{S}}$ instead of multisets, with the constraint that $b(p) \in \mathbb{N}$ for any regular place $p \in P$ and $b(\mathbf{p}) \in \{0, \infty\}$ for any persistent place $\mathbf{p} \in \mathbf{P}$. To guarantee that this property is preserved by firing sequences, we assume that the post-set $t^\bullet$ of a transition t is the bag such that: $(t^\bullet)(p) = 1$ if $(t, p) \in F$ (as usual); $(t^\bullet)(\mathbf{p}) = \infty$ if $(t, \mathbf{p}) \in F$; and $(t^\bullet)(s) = 0$ if $(t, s) \notin F$. We say that a transition t is *persistent* if it is attached to persistent places only (i.e. if ${}^\bullet t \cup t^\bullet \subseteq \mathbf{P}$).

The notions of enabling, firing, firing sequence and reachability extend in the obvious way to p-nets (when markings are replaced by bags). For example, a transition t is *enabled* at the bag b, written $b \xrightarrow{t}$, if ${}^\bullet t \subseteq b$, and the firing of an enabled transitions is written $b \xrightarrow{t} b'$ with $b' = (b \setminus {}^\bullet t) \cup t^\bullet$.

A firing sequence is *stuttering* if it has multiple occurrences of a persistent transition. Since firing a persistent transition t multiple times is inessential, we consider non-stuttering firing sequences. (Alternatively, we can add a marked regular place p_t to the preset of each persistent transition t, so t fires at most once.)

A marked p-net (N, b_0) is *1-∞-safe* if each reachable bag $b \in [b_0\rangle$ is such that $b(p) \in 2$ for all $p \in P$ and $b(\mathbf{p}) \in \{0, \infty\}$ for all $\mathbf{p} \in \mathbf{P}$. Note that in 1-∞-safe nets the amount of information conveyed by any reachable bag is finite, as each place is associated with one bit of information (marked or unmarked). Graphically, persistent places are represented by circles with double border (and they are either empty or contain a single token). See Fig. 3c for an example.

The notion of confusion extends to p-nets, by checking direct conflicts w.r.t. regular places only.

2.5 Dynamic Nets

Dynamic nets [Asperti and Busi 2009] are Petri nets whose sets of places and transitions may increase dynamically. We focus on a subclass of persistent dynamic nets that only allows for changes in the set of transitions, which is defined as follows.

Definition 2.1 (Dynamic p-nets)**.** The set $\textsc{dn}(\mathbb{S})$ is the *least set* satisfying the recursive equation:

$$\textsc{dn}(\mathbb{S}) = \{(T, b) \mid T \subseteq 2^{\mathbb{S}} \times \textsc{dn}(\mathbb{S}) \ \wedge \ T \text{ finite} \ \wedge \ b \in \mathbb{N}_\infty^{\mathbb{S}}\}$$

The definition above is a domain equation for the set of dynamic p-nets over the set of places $\mathbb{S}$: the set $\text{DN}(\mathbb{S})$ is the least fixed point of the equation. The simplest elements in $\text{DN}(\mathbb{S})$ are pairs $(\emptyset, b)$ with bag $b \in \mathbb{N}_\infty^{\mathbb{S}}$ (with $b(p) \in \mathbb{N}$ for any $p \in P$ and $b(\mathbf{p}) \in \{0, \infty\}$ for any $\mathbf{p} \in \mathbf{P}$). Nets (T, b) are defined recursively; indeed any element $t = (S, N) \in T$ stands for a transition with preset S and postset N, which is another element of $\text{DN}(\mathbb{S})$. An ordinary transition from b to b' has thus the form $(b, (\emptyset, b'))$. We write $S \to N$ for the transition $t = (S, N)$, ${}^\bullet t = S$ for its *preset*, and $t^\bullet = N \in \text{DN}(\mathbb{S})$ for its *postset*. For $N = (T, b)$ we say that T is the set of *top transitions* of N. All the other transitions are called dynamic.

The firing rule rewrites a dynamic p-net (T, b) to another one. The firing of a transition $t = S \to (T', b') \in T$ consumes the preset S and releases both the transitions T' and the tokens in b'. Formally, if $t = S \to (T', b') \in T$ with $S \subseteq b$ then $(T, b) \xrightarrow{t} (T \cup T', (b \setminus S) \cup b')$.

The notion of 1-∞-safe dynamic p-net is defined analogously to p-nets by considering the bags b of reachable states (T, b).

A sample of a dynamic net is shown in Fig. 4a, whose only dynamic transition, which is activated by t_3, is depicted with dashed border. The arrow between t_3 and b denotes the fact that b is activated dynamically by the firing of $t_3 : \overline{3} \to (\{b : 2 \to 4\}, \{\overline{5}\})$.

We show that any dynamic p-net can be encoded as a (flat) p-net. Our encoding resembles the one in [Asperti and Busi 2009], but it is simpler because we do not need to handle place creation. Intuitively, we release any transition t immediately but we add a persistent place $\mathbf{p}_t$ to its preset, to enable t dynamically ($\mathbf{p}_t$ is initially empty iff t is not a top transition). Given a set T of transitions, b_T is the bag such that $b_T(\mathbf{p}_t) = \infty$ if $t \in T$ and $b_T(s) = 0$ otherwise.

For $N = (T, b) \in \text{DN}(\mathbb{S})$, we let $\mathbb{T}(N) = T \cup \bigcup_{t \in T} \mathbb{T}(t^\bullet)$ be the set of all (possibly nested) transitions appearing in N. From Definition 2.1 it follows that $\mathbb{T}(N)$ is finite and well-defined.

Definition 2.2 (From dynamic to static)**.** Given $N = (T, b) \in \text{DN}(\mathbb{S})$, the corresponding p-net $(\!|N|\!)$ is defined as $(\!|N|\!) = (\mathbb{S} \cup \mathbf{P}_{\mathbb{T}(N)}, \mathbb{T}(N), F, b \cup b_T)$, where

- $\mathbf{P}_{\mathbb{T}(N)} = \{\mathbf{p}_t \mid t \in \mathbb{T}(N)\}$; and
- F is such that for any $t = S \to (T', b') \in \mathbb{T}(N)$ then $t : {}^\bullet t \cup \{\mathbf{p}_t\} \to b' \cup b_{T'}$.

The transitions of $(\!|N|\!)$ are those from N (set $\mathbb{T}(N)$). Any place of N is also a place of $(\!|N|\!)$ (set $\mathbb{S}$). In addition, there is one persistent place $\mathbf{p}_t$ for each $t \in \mathbb{T}(N)$ (set $\mathbf{P}_{\mathbb{T}(N)}$). The initial marking of $(\!|N|\!)$ is that of N (i.e., b) together with the persistent tokens that enable the top transitions of N (i.e., b_T). Adding b_T is convenient for the statement in Proposition 2.4, but we could safely remove $\mathbf{P}_T \subseteq \mathbf{P}_{\mathbb{T}(N)}$ (and b_T) from the flat p-net without any consequence.

Example 2.3. The dynamic p-net N in Fig. 4a is encoded as the p-net $(\!|N|\!)$ in Fig. 4b, which has as many transitions as N, but the preset of every transition contains an additional persistent place (depicted in grey) to indicate transition's availability. All the new places but p_b are marked because the corresponding transitions are initially available. Contrastingly, p_b is unmarked because the corresponding transition becomes available after the firing of t_3.

The following result shows that all computations of a dynamic p-net can be mimicked by the corresponding p-net and vice versa. Hence, the encoding preserves also 1-safety over regular places.

Proposition 2.4. *Let $N = (T, b) \in \text{DN}(\mathbb{S})$. Then,*

1. *$N \xrightarrow{t} N'$ implies $(\!|N|\!) \xrightarrow{t} (\!|N'|\!)$;*
2. *Moreover, $(\!|N|\!) \xrightarrow{t} N'$ implies there exists N'' such that $N \xrightarrow{t} N''$ and $N' = (\!|N''|\!)$.*

Corollary 2.5. *$(\!|N|\!)$ is 1-∞-safe iff N is 1-safe.*

3 From Petri Nets to Dynamic P-Nets

In this section we show that any (finite, acyclic) net N can be associated with a confusion-free, dynamic p-net $[\![N]\!]$ by suitably encoding loci of decision. The mapping builds on the structural cell decomposition introduced below.

3.1 Structural Branching Cells

A structural branching cell represents a statically determined locus of choice, where the firing of some transitions is considered against all the possible conflicting alternatives. To each transition t we assign an s-cell $[t]$. This is achieved by taking the equivalence class of t w.r.t. the equivalence relation $\leftrightarrow$ induced by the least preorder $\sqsubseteq$ that includes immediate conflict $\#_0$ and causality $\preceq$. For convenience, each s-cell $[t]$ also includes the places in the presets of the transitions in $[t]$, i.e., we let the relation Pre^{-1} be also included in $\sqsubseteq$, with $\text{Pre} = F \cap (P \times T)$. This way, if $(p, t) \in F$ then $p \sqsubseteq t$ because $p \preceq t$ and $t \sqsubseteq p$ because $(t, p) \in \text{Pre}^{-1}$. Formally, we let $\sqsubseteq$ be the transitive closure of the relation $\#_0 \cup \preceq \cup \text{Pre}^{-1}$. Since $\#_0$ is subsumed by the transitive closure of the relation $\preceq \cup \text{Pre}^{-1}$, we equivalently set $\sqsubseteq = (\preceq \cup \text{Pre}^{-1})^*$. Then, we let $\leftrightarrow = \{(x, y) \mid x \sqsubseteq y \wedge y \sqsubseteq x\}$. Intuitively, the choices available in the equivalence class $[t]_{|\leftrightarrow}$ of a transition t must be resolved atomically.

Definition 3.1 (S-cells)**.** Let $N = (P, T, F)$ be a finite, nondeterministic occurrence net. The set $\text{BC}(N)$ of *s-cells* is the set of equivalence classes of $\leftrightarrow$, i.e., $\text{BC}(N) = \{[t]_{|\leftrightarrow} \mid t \in T\}$.

We let $\mathbb{C}$ range over s-cells. By definition it follows that for all $\mathbb{C}, \mathbb{C}' \in \text{BC}(N)$, if $\mathbb{C} \cap \mathbb{C}' \neq \emptyset$ then $\mathbb{C} = \mathbb{C}'$. For any s-cell $\mathbb{C}$, we denote by $N_\mathbb{C}$ the subnet of N whose elements are in $\mathbb{C} \cup \bigcup_{t \in \mathbb{C}} t^\bullet$. Abusing the notation, we denote by ${}^\circ\mathbb{C}$ the set of all the initial places in $N_\mathbb{C}$ and by $\mathbb{C}^\circ$ the set of all the final places in $N_\mathbb{C}$.

Definition 3.2 (Transactions)**.** Let $\mathbb{C} \in \text{BC}(N)$. Then, a *transaction* θ of $\mathbb{C}$, written $\theta : \mathbb{C}$, is a maximal (deterministic) process of $N_\mathbb{C}$.

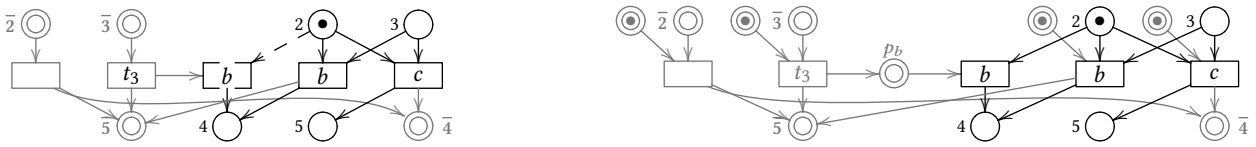
Since the set of transitions in a transaction θ uniquely determines the corresponding process in $N_\mathbb{C}$, we write a transaction θ simply as the set of its transitions.

Example 3.3. The net N in Fig. 3a has the three s-cells shown in Fig. 5a, whose transactions are listed in Fig. 5b. For $\mathbb{C}_1$ and $\mathbb{C}_2$, each transition defines a transaction; $\mathbb{C}_3$ has one transaction associated with c and one with (the concurrent firing of) b and g.

The following operation $\ominus$ is instrumental for the definition of our encoding and stands for the removal of a minimal place of a net and all the elements that causally depend on it. Formally, $N \ominus p$ is the least set that satisfies the rules (where ${}^\circ(\_)$ has higher precedence over set difference):

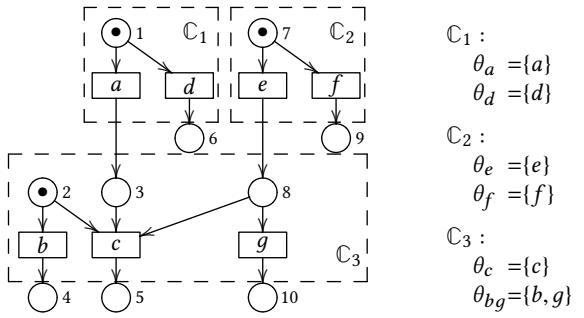
$$\frac{q \in {}^\circ N \setminus \{p\}}{q \in N \ominus p} \qquad \frac{t \in N \qquad {}^\bullet t \subseteq N \ominus p}{t \in N \ominus p} \qquad \frac{t \in N \ominus p \qquad q \in t^\bullet}{q \in N \ominus p}$$

Example 3.4. Consider the s-cells in Fig. 5a. The net $N_{\mathbb{C}_1} \ominus 1$ is empty because every node in $N_{\mathbb{C}_1}$ causally depends on 1. Similarly, $N_{\mathbb{C}_2} \ominus 7$ is empty. The cases for $\mathbb{C}_3$ are in Figs. 5c–5e.



(a) A dynamic p-net N **(b)** The corresponding p-net $(\!|N|\!)$

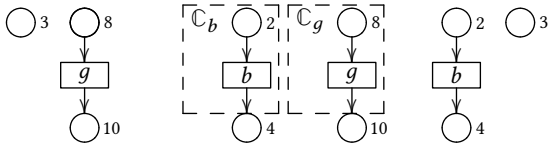
Figure 4. A dynamic p-net encoded as a p-net



(a) Structural branching cells

$\mathbb{C}_1$:
$\theta_a = \{a\}$
$\theta_d = \{d\}$
$\mathbb{C}_2$:
$\theta_e = \{e\}$
$\theta_f = \{f\}$
$\mathbb{C}_3$:
$\theta_c = \{c\}$
$\theta_{bg} = \{b, g\}$

(b) Transactions



(c) $N_{\mathbb{C}_3} \ominus 2$ **(d)** $N_{\mathbb{C}_3} \ominus 3$ **(e)** $N_{\mathbb{C}_3} \ominus 8$

Figure 5. Structural branching cells (running example)

3.2 Encoding s-cells as confusion-free dynamic nets

Intuitively, the proposed encoding works by explicitly representing the fact that a place will not be marked in a computation. We denote with $\overline{\mathbf{p}}$ the place that models such "negative" information about the regular place p and let $\overline{\mathbf{P}} = \{\overline{\mathbf{p}} \mid p \in P\}$.[1] The encoding uses negative information to recursively decompose s-cells under the assumption that some of their minimal places will stay empty.

Definition 3.5 (From s-cells to dynamic p-nets)**.** Let $N = (P, T, F, m)$ be a marked occurrence net. Its dynamic p-net $[\![N]\!] \in \text{DN}(P \cup \overline{\mathbf{P}})$ is defined as $[\![N]\!] = (T_{\text{pos}} \cup T_{\text{neg}}, m)$, where:

$$T_{\text{pos}} = \{\ {}^\circ\mathbb{C} \to (\emptyset, \theta^\circ \cup \overline{\mathbb{C}^\circ \setminus \theta^\circ})\ \mid\ \mathbb{C} \in \text{BC}(N) \text{ and } \theta : \mathbb{C}\ \}$$
$$T_{\text{neg}} = \{\ \overline{\mathbf{p}} \to (T', \overline{\mathbb{C}^\circ \setminus (N_{\mathbb{C}} \ominus p)^\circ})\ \mid\ \mathbb{C} \in \text{BC}(N) \text{ and } p \in {}^\circ\mathbb{C} \text{ and } (T', b) = [\![N_{\mathbb{C}} \ominus p]\!]\ \}$$

For any s-cell $\mathbb{C}$ of N and transaction $\theta : \mathbb{C}$, the encoding generates a transition $t_{\theta,\mathbb{C}} = ({}^\circ\mathbb{C} \to (\emptyset, \theta^\circ \cup \overline{\mathbb{C}^\circ \setminus \theta^\circ})) \in T_{\text{pos}}$ to mimic the atomic execution of θ. Despite ${}^\circ\theta$ may be strictly included in ${}^\circ\mathbb{C}$, we set ${}^\circ\mathbb{C}$ as the preset of $t_{\theta,\mathbb{C}}$ to ensure that the execution of θ only starts when the whole s-cell $\mathbb{C}$ is enabled. Each transition $t_{\theta,\mathbb{C}} \in T_{\text{pos}}$ is a transition of an ordinary Petri net because its postset consists of (i) the final places of θ and (ii) the negative versions of the places in $\mathbb{C}^\circ \setminus \theta^\circ$. A token in $\overline{\mathbf{p}} \in \overline{\mathbb{C}^\circ \setminus \theta^\circ}$ represents the fact that the corresponding ordinary place $p \in \mathbb{C}^\circ$ will not be marked because it depends on discarded transitions (not in θ).

[1] The notation $\overline{P}$ denotes just a set of places whose names are decorated with a bar; it should not be confused with usual set complement.

$t_a : 1 \to (\emptyset, \{3, \overline{6}\})$ for θ_a
$t_d : 1 \to (\emptyset, \{6, \overline{3}\})$ for θ_d
$t_1 : \overline{1} \to (\emptyset, \{\overline{3}, \overline{6}\})$
$t_e : 7 \to (\emptyset, \{8, \overline{9}\})$ for θ_e
$t_f : 7 \to (\emptyset, \{9, \overline{8}\})$ for θ_f
$t_7 : \overline{7} \to (\emptyset, \{\overline{8}, \overline{9}\})$
$t_{bg} : 2, 3, 8 \to (\emptyset, \{4, 10, \overline{5}\})$ for θ_{bg}
$t_c : 2, 3, 8 \to (\emptyset, \{5, \overline{4}, \overline{10}\})$ for θ_c

$t_2 : \overline{2} \to (\{t_g, t'_8\}, \{\overline{4}, \overline{5}\})$
$t_3 : \overline{3} \to (\{t_b, t'_2, t_g, t'_8\}, \{\overline{5}\})$
$t_8 : \overline{8} \to (\{t_b, t'_2\}\{\overline{5}, \overline{10}\})$

where
$t_b : 2 \to (\emptyset, \{4\})$
$t'_2 : \overline{2} \to (\emptyset, \{\overline{4}\})$
$t_g : 8 \to (\emptyset, \{10\})$
$t'_8 : \overline{8} \to (\emptyset, \{\overline{10}\})$

Figure 6. Encoding of branching cells (running example)

Negative information is propagated by the transitions in T_{neg}. For each cell $\mathbb{C}$ and place $p \in {}^\circ\mathbb{C}$, there exists one dynamic transition $t_{p,\mathbb{C}} = \overline{\mathbf{p}} \to (T', \overline{\mathbb{C}^\circ \setminus (N_{\mathbb{C}} \ominus p)^\circ})$ whose preset is just $\overline{\mathbf{p}}$ and whose postset is defined in terms of the subnet $N_{\mathbb{C}} \ominus p$. The postset of $t_{p,\mathbb{C}}$ accounts for two effects of propagation: (i) the generation of the negative tokens for all maximal places of $\mathbb{C}$ that causally depend on p, i.e., for the negative places associated with the ones in $\mathbb{C}^\circ$ that are not in $(N_{\mathbb{C}} \ominus p)^\circ$; and (ii) the activation of all transitions T' obtained by encoding $N_{\mathbb{C}} \ominus p$, i.e., the behaviour of the branching cell $\mathbb{C}$ after the token in the minimal place p is excluded. We remark that the bag b in $(T', b) = [\![N_{\mathbb{C}} \ominus p]\!]$ is always empty, because i) $N_{\mathbb{C}}$ is unmarked and, consequently, $N_{\mathbb{C}} \ominus p$ is unmarked, and ii) the initial marking of $[\![N]\!]$ corresponds to the initial marking of N.

Example 3.6. Consider the net N and its s-cells in Fig. 5a. Then, $[\![N]\!] = (T, b)$ is defined such that b is the initial marking of N, i.e., $b = \{1, 2, 7\}$, and T has the transitions shown in Fig. 6.

First consider the s-cell $\mathbb{C}_1$. T_{pos} contains one transition for each transaction in $\mathbb{C}_1$, namely t_a (for $\theta_a : \mathbb{C}_1$) and t_d (for $\theta_d : \mathbb{C}_1$). Both t_a and t_d have ${}^\circ\mathbb{C}_1 = \{1\}$ as preset. By definition of T_{pos}, both transitions have empty sets of transitions in their postsets. Additionally, $t_a^\bullet$ produces tokens in $\theta_a^\circ = \{3\}$ (positive) and $\overline{\mathbb{C}_1^\circ \setminus \theta_a^\circ} = \overline{\{3, 6\} \setminus \{3\}} = \{\overline{6}\}$ (negative), while $t_d^\bullet$ produces tokens in $\theta_d^\circ = \{6\}$ and $\overline{\mathbb{C}_1^\circ \setminus \theta_d^\circ} = \{\overline{3}\}$. Finally, $t_1 \in T_{\text{neg}}$ propagates negative tokens for the unique place in ${}^\circ\mathbb{C}_1 = \{1\}$. Since $N_{\mathbb{C}_1} \ominus 1$ is the empty net, $[\![N_{\mathbb{C}_1} \ominus 1]\!] = (\emptyset, \emptyset)$. Hence, t_1 produces negative tokens for all maximal places of $\mathbb{C}_1$, i.e., $\{\overline{3}, \overline{6}\}$. For the s-cell $\mathbb{C}_2$ we analogously obtain the transitions t_e, t_f and t_7.

The s-cell $\mathbb{C}_3$ has two transactions θ_{bg} and θ_c. Hence, $[\![N]\!]$ has two transitions $t_{bg}, t_c \in T_{\text{pos}}$. Despite θ_{bg} mimics the firing of b and g, which are disconnected from the place 3, it is included in the preset of t_{bg} to postpone the firing of t_{bg} until $\mathbb{C}_1$ is executed. Transitions $t_2, t_3, t_8 \in T_{\text{neg}}$ propagate the negative information for

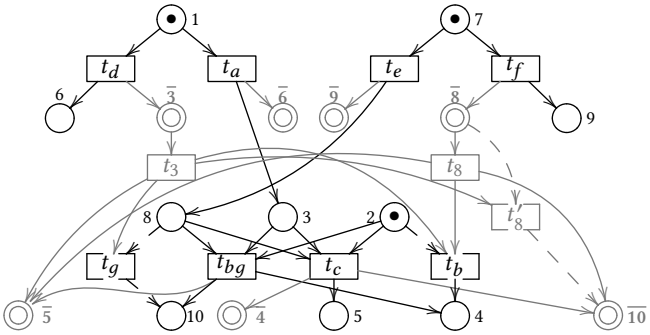


Figure 7. Dynamic net $[\![N]\!]$ (running example)

the places in ${}^\circ\mathbb{C}_3 = \{2, 3, 8\}$. The transition t_3 has ${}^\bullet t_3 = \{\overline{3}\}$ as its preset and its postset is obtained from $N_{\mathbb{C}_3} \ominus 3$, which has two (sub) s-cells $\mathbb{C}_b$ and $\mathbb{C}_g$ (see Fig. 5d). The transitions t_b and t'_2 arise from $\mathbb{C}_b$, and t_g and t'_8 from $\mathbb{C}_g$. Hence, $t_3^\bullet = (\{t_b, t'_2, t_g, t'_8\}, \{\overline{5}\})$ because $[\![N_{\mathbb{C}_3} \ominus 3]\!] = (\{t_b, t'_2, t_g, t'_8\}, \emptyset)$ and $\overline{\mathbb{C}_3^\circ \setminus (N_{\mathbb{C}_3} \ominus 3)^\circ} = \{\overline{5}\}$. Similarly, we derive t_2 from $N_{\mathbb{C}_3} \ominus 2$ and t_8 from $N_{\mathbb{C}_3} \ominus 8$.

We now highlight some features of the encoded net. First, the set of top transitions is free-choice: positive and negative transitions have disjoint presets and the presets of any two positive transitions either coincide (if they arise from the same s-cell) or are disjoint. Recursively, this property holds at any level of nesting. Hence, the only source of potential confusion is due to the combination of top transitions and those activated dynamically, e.g., t_b and either t_{bg} or t_c. However, t_b is activated only when either $\overline{3}$ or $\overline{8}$ are marked, while ${}^\bullet t_{bg} = {}^\bullet t_c = \{2, 3, 8\}$. Then, confusion is avoided if p and $\overline{\mathbf{p}}$ can never be marked in the same execution (Lemma 3.7).

The net $[\![N]\!]$ is shown in Fig. 7, where the places $\{\overline{\mathbf{1}}, \overline{\mathbf{2}}, \overline{\mathbf{7}}\}$ and the transitions $\{t_1, t_7, t_2, t'_2\}$ are omitted because superseded by the initial marking $\{1, 2, 7\}$.

We remark that the same dynamic transition can be released by the firing of different transitions (e.g., t_b by t_3 and t_8) and possibly several times in the same computation. Similarly, the same negative information can be generated multiple times. However this duplication has no effect, since we handle persistent tokens. For instance, the firing sequence $t_d; t_f; t_3; t_8$ releases two copies of t_b and marks $\overline{5}$ twice. This is inessential for reachability, but has interesting consequences w.r.t. causal dependencies (see Section 5).

We now show that the encoding generates confusion-free nets. We start by stating a useful property of the encoding that ensures that an execution cannot generate tokens in both p and $\overline{\mathbf{p}}$.

Lemma 3.7 (Negative and positive tokens are in exclusion)**.** *If $[\![N]\!] \to^* (T, b)$ and $\overline{\mathbf{p}} \in b$ then $(T, b) \to^* (T', b')$ implies that $p \notin b'$.*

We now observe from Def. 3.5 that for any transition $t \in [\![N]\!] \in \text{DN}(P \cup \overline{\mathbf{P}})$ it holds that either ${}^\bullet t \subseteq P$ or ${}^\bullet t \subseteq \overline{\mathbf{P}}$. The next result says that whenever there exist two transitions t and t' that have different but overlapping presets, at least one of them is disabled by the presence of a negative token in the marking b.

Lemma 3.8 (Nested rules do not collide)**.** *Let $[\![N]\!] \in \text{DN}(P \cup \overline{\mathbf{P}})$. If $[\![N]\!] \to^* (T, b)$ then for all $t, t' \in T$ s.t. ${}^\bullet t \neq {}^\bullet t'$ and ${}^\bullet t \cap {}^\bullet t' \cap P \neq \emptyset$ it holds that there is $p \in P \cap ({}^\bullet t \cup {}^\bullet t')$ such that $\overline{\mathbf{p}} \in b$.*

The main result states that $[\![\cdot]\!]$ generates confusion-free nets.

Theorem 3.9. *Let $[\![N]\!] \in \text{DN}(P \cup \overline{\mathbf{P}})$. If $[\![N]\!] \to^* (T, b) \xrightarrow{t}$ and $(T, b) \xrightarrow{t'}$ then either ${}^\bullet t = {}^\bullet t'$ or ${}^\bullet t \cap {}^\bullet t' = \emptyset$.*

Corollary 3.10. *Any net $[\![N]\!] \in \text{DN}(P \cup \overline{\mathbf{P}})$ is confusion-free.*

Finally, we can combine the encoding $[\![\cdot]\!]$ with $(\!|\cdot|\!)$ (from Section 2.5) to obtain a (flat) 1-∞-safe, confusion-free, p-net $(\!|[\![N]\!]|\!)$, that we call the *uniformed net* of N. By Proposition 2.4 we get that the uniformed net $(\!|[\![N]\!]|\!)$ is also confusion-free by construction.

Corollary 3.11. *Any p-net $(\!|[\![N]\!]|\!)$ is confusion-free.*

4 Static vs Dynamic cell decomposition

As mentioned in the Introduction, Abbes and Benveniste proposed a way to remove confusion by dynamically decomposing prime event structures. In Sections 4.1 and 4.2 we recall the basics of the AB's approach as introduced in Abbes and Benveniste [2005, 2006, 2008]. Then, we show that there is an operational correspondence between AB decomposition and s-cells introduced in Section 3.1.

4.1 Prime Event Structures

A *prime event structure* (also *PES*) [Nielsen et al. 1981; Winskel 1987] is a triple $\mathcal{E} = (E, \preceq, \#)$ where: E is the set of *events*; the *causality relation* $\preceq$ is a partial order on events; the *conflict relation* $\#$ is a symmetric, irreflexive relation on events such that conflicts are inherited by causality, i.e., $\forall e_1, e_2, e_3 \in E.\ e_1 \# e_2 \preceq e_3 \Rightarrow e_1 \# e_3$.

The PES $\mathcal{E}_N$ associated with a net N can be formalised using category theory as a chain of universal constructions, called coreflections. Hence, for each PES $\mathcal{E}$, there is a standard, unique (up to isomorphism) nondeterministic occurrence net $N_\mathcal{E}$ that yields $\mathcal{E}$ and thus we can freely move from one setting to the other.

Consider the nets in Figs. 1a and 3a. The corresponding PESs are shown below each net. Events are in bijective correspondence with the transitions of the nets. Strict causality is depicted by arrows and immediate conflict by curly lines.

Given an event e, its *downward closure* $\lfloor e \rfloor = \{e' \in E \mid e' \preceq e\}$ is the set of causes of e. As usual, we assume that $\lfloor e \rfloor$ is finite for any e. Given $B \subseteq E$, we say that B is *downward closed* if $\forall e \in B.\ \lfloor e \rfloor \subseteq B$ and that B is *conflict-free* if $\forall e, e' \in B.\ \neg(e \# e')$. We let the *immediate conflict* relation $\#_0$ be defined on events by letting $e \#_0 e'$ iff $(\lfloor e \rfloor \times \lfloor e' \rfloor) \cap \# = \{(e, e')\}$, i.e., two events are in immediate conflict if they are in conflict but their causes are compatible.

4.2 Abbes and Benveniste's Branching Cells

In the following we assume that a (finite) PES $\mathcal{E} = (E, \preceq, \#)$ is given. A *prefix* $B \subseteq E$ is any downward-closed set of events (possibly with conflicts). Any prefix B induces an event structure $\mathcal{E}_B = (B, \preceq_B, \#_B)$ where $\preceq_B$ and $\#_B$ are the restrictions of $\preceq$ and $\#$ to the events in B. A *stopping prefix* is a prefix B that is closed under immediate conflicts, i.e., $\forall e \in B, e' \in E.\ e \#_0 e' \Rightarrow e' \in B$. Intuitively, a stopping prefix is a prefix whose (immediate) choices are all available. It is *initial* if the only stopping prefix strictly included in B is $\emptyset$. We assume that any $e \in E$ is contained in a finite stopping prefix.

A *configuration* $v \subseteq \mathcal{E}$ is any set of events that is downward closed and conflict-free. Intuitively, a configuration represents (the state reached after executing) a concurrent but deterministic computation of $\mathcal{E}$. Configurations are ordered by inclusion and we denote by $\mathcal{V}_\mathcal{E}$ the poset of finite configurations of $\mathcal{E}$ and by $\Omega_\mathcal{E}$ the poset of maximal configurations of $\mathcal{E}$.

The *future* of a configuration v, written E^v, is the set of events that can be executed after v, i.e., $E^v = \{e \in E \setminus v \mid \forall e' \in v. \neg(e \# e')\}$. We write $\mathcal{E}^v$ for the event structure induced by E^v. We assume that any finite configuration enables only finitely many events, i.e., the set of minimal elements in E^v w.r.t. $\leq$ is finite for any $v \in \mathcal{V}_{\mathcal{E}}$.

A configuration v is *stopped* if there is a stopping prefix B with $v \in \Omega_B$. and v is *recursively stopped* if there is a finite sequence of configurations $\emptyset = v_0 \subset \ldots \subset v_n = v$ such that for any $i \in [0, n)$ the set $v_{i+1} \setminus v_i$ is a finite stopped configuration of $\mathcal{E}^{v_i}$ for v_i in $\mathcal{E}$.

A *branching cell* is any initial stopping prefix of the future $\mathcal{E}^v$ of a finite recursively stopped configuration v. Intuitively, a branching cell is a minimal subset of events closed under immediate conflict. We remark that branching cells are determined by considering the whole (future of the) event structure $\mathcal{E}$ and they are recursively computed as $\mathcal{E}$ is executed. Remarkably, every maximal configuration has a branching cell decomposition.

Example 4.1. Consider the PES $\mathcal{E}_N$ in Fig. 3a and its maximal configuration $v = \{a, e, b, g\}$. We show that v is recursively stopped by exhibiting a branching cell decomposition. The initial stopping prefixes of $\mathcal{E}_N = \mathcal{E}_N^{\emptyset}$ are shown in Fig. 8a. There are two possibilities for choosing $v_1 \subseteq v$ and v_1 recursively stopped: either $v_1 = \{a\}$ or $v_1 = \{e\}$. When $v_1 = \{a\}$, the choices for v_2 are determined by the stopping prefixes of $\mathcal{E}_N^{\{a\}}$ (see Fig. 8b) and the only possibility is $v_2 = \{a, e\}$. From $\mathcal{E}_N^{\{a,e\}}$ in Fig. 8c, we take $v_3 = v$. Note that $\{a, e, b\}$ is not recursively stopped because $\{b\}$ is not maximal in the stopping prefix of $\mathcal{E}_N^{\{a,e\}}$ (see Fig. 8c). Finally, note that the branching cells $\mathcal{E}_N^{\{a\}}$ (Fig. 8b) and $\mathcal{E}_N^{\{d\}}$ (Fig. 8d) correspond to different choices in $\mathcal{E}_N^{\emptyset}$ and thus have different stopping prefixes.

4.3 Relating s-cells and AB's decomposition

The recursively stopped configurations of a net N characterise all the allowed executions of N. Hence, we formally link the recursively stopped configurations of $\mathcal{E}_N$ with the computations of the uniformed net $(\!|[\![N]\!]|\!)$. For technical convenience, we first show that the recursively stopped configurations of $\mathcal{E}_N$ are in one-to-one correspondence with the computations of the dynamic net $[\![N]\!]$. Then, the desired correspondence is obtained by using Proposition 2.4 to relate the computations of a dynamic net and its associated p-net.

We rely on the auxiliary map $\|-\|$ that links transitions in $[\![N]\!]$ with events in $\mathcal{E}_N$. Specifically, $\|-\|$ associates each transition t of $[\![N]\!]$ with the set $\|t\|$ of transitions of N (also events in $\mathcal{E}_N$) that are encoded by t. Formally,

$$\|t\| = \begin{cases} ev(\theta) & \text{if } t = t_{\theta,\mathbb{C}} \in T_{\text{pos}} \\ \emptyset & \text{if } t \in T_{\text{neg}} \end{cases}$$

where $ev(\theta)$ is the set of transitions in θ.

Example 4.2. Consider the net N in Fig. 5a which is encoded as the dynamic p-net in Fig. 6. The auxiliary mapping $\|\_\|$ is as follows

$$\|t_a\| = \{a\} \quad \|t_d\| = \{d\} \quad \|t_e\| = \{e\} \quad \|t_f\| = \{f\}$$
$$\|t_{bg}\| = \{b, g\} \quad \|t_c\| = \{c\} \quad \|t_b\| = \{b\} \quad \|t_g\| = \{g\}$$
$$\|t\| = \emptyset \text{ if } t \in \{t_1, t_7, t_2, t_3, t_8, t_2', t_8'\}$$

A transition $t_{\theta,\mathbb{C}}$ of $(\!|N|\!)$ associated with a transaction $\theta : \mathbb{C}$ of N is mapped to the transitions of θ. For instance, t_a is mapped to $\{a\}$, which is the only transition in θ_a. Differently, transitions that propagate negative information, i.e., $t \in \{t_1, t_7, t_2, t_3, t_8, t_2', t_8'\}$, are mapped to $\emptyset$ because they do not encode any transition of N.

In what follows we write $M \Longrightarrow M'$ for a possibly empty firing sequence $M \xrightarrow{t_1 \cdots t_n} M'$ such that $\|t_i\| = \emptyset$ for all $i \in [1, n]$. If $\|t\| \neq \emptyset$, we write $M \overset{t}{\Longrightarrow} M'$ if $M \Longrightarrow M_0 \overset{t}{\Longrightarrow} M_1 \Longrightarrow M'$ for some M_0, M_1. Moreover, we write $M \xRightarrow{t_1 \cdots t_n}$ if there exist $M_1, ..., M_n$ such that $M \overset{t_1}{\Longrightarrow} M_1 \overset{t_2}{\Longrightarrow} \cdots \overset{t_n}{\Longrightarrow} M_n$.

The following result states that the computations of any dynamic p-net produced by $[\![\_]\!]$ are in one-to-one correspondence with the recursively stopped configurations of Abbes and Benveniste.

Lemma 4.3. *Let N be an occurrence net.*

1. *If $[\![N]\!] \xRightarrow{t_1 \cdots t_n}$, then $v = \bigcup_{1 \leq i \leq n} \|t_i\|$ is recursively stopped in $\mathcal{E}_N$ and $(\|t_i\|)_{1 \leq i \leq n}$ is a valid decomposition of v.*
2. *If v is recursively stopped in $\mathcal{E}_N$, then for any valid decomposition $(v_i)_{1 \leq i \leq n}$ there exists $[\![N]\!] \xRightarrow{t_1 \cdots t_n}$ such that $\|t_i\| = v_i$.*

Example 4.4. Consider the branching cell decomposition for $v = \{a, e, b, g\} \in \mathcal{E}_v$ discussed in Ex. 4.1. Then, the net $[\![N]\!]$ in Ex. 3.6 can mimic that decomposition with the following computation

$$(T, \{1, 2, 7\}) \xrightarrow{t_a} (T, \{2, 3, 7, \overline{6}\}) \xrightarrow{t_e} (T, \{2, 3, 8, \overline{6}, \overline{9}\}) \xrightarrow{t_{bg}} (T, \{4, 10, \overline{5}, \overline{6}, \overline{9}\})$$

with $v_1 = \|t_a\| = \{a\}$, $v_2 = \|t_e\| = \{e\}$, and $v_3 = \|t_{bg}\| = \{b, g\}$.

From Lemma 4.3 and Proposition 2.4 we obtain the next result.

Theorem 4.5 (Correspondence)**.** *Let N be an occurrence net.*

1. *If $(\!|[\![N]\!]|\!) \xRightarrow{t_1 \cdots t_n}$, then $v = \bigcup_{1 \leq i \leq n} \|t_i\|$ is recursively stopped in $\mathcal{E}_N$ and $(\|t_i\|)_{1 \leq i \leq n}$ is a valid decomposition of v.*
2. *If v is recursively stopped in $\mathcal{E}_N$, then for any valid decomposition $(v_i)_{1 \leq i \leq n}$ there exists $(\!|[\![N]\!]|\!) \xRightarrow{t_1 \cdots t_n}$ such that $\|t_i\| = v_i$.*

By (1) above, any computation of $(\!|[\![N]\!]|\!)$ corresponds to a (recursively stopped) configuration of $\mathcal{E}_N$, i.e., a process of N. By (2), every execution of N that can be decomposed in terms of AB's branching cells is preserved by $(\!|[\![N]\!]|\!)$, because any recursively stopped configuration of $\mathcal{E}_N$ is mimicked by $(\!|[\![N]\!]|\!)$.

5 Concurrency of the Uniformed Net

In this section we study the amount of concurrency still present in the uniformed net $(\!|[\![N]\!]|\!)$. Here, we extend the notion of a process to the case of 1-∞-safe p-nets and we show that all the *legal* firing sequences of a process of the uniformed net $(\!|[\![N]\!]|\!)$ are executable.

The notion of deterministic occurrence net is extended to p-nets by slightly changing the definitions of conflict and causal dependency: (i) two transitions are *not* in conflict when all shared places are persistent, (ii) a persistent place can have more than one immediate cause in its preset, which introduces OR-dependencies.

Definition 5.1 (Persistent process)**.** An *occurrence p-net* $O = (P \cup \mathbf{P}, T, F)$ is an acyclic p-net such that $|p^\bullet| \leq 1$ and $|{}^\bullet p| \leq 1$ for any $p \in P$ (but not necessarily for those in $\mathbf{P}$).

A *persistent process* for N is an occurrence p-net O together with a net morphism $\pi : O \to N$ that preserves presets and postsets and the distinction between regular and persistent places. Without loss of generality, when N is acyclic, we assume that O is a subnet of N (with the same initial marking) and π is the identity.

In an ordinary occurrence net, the causes of an item x are all its predecessors. In p-nets, the alternative sets of causes of an item x

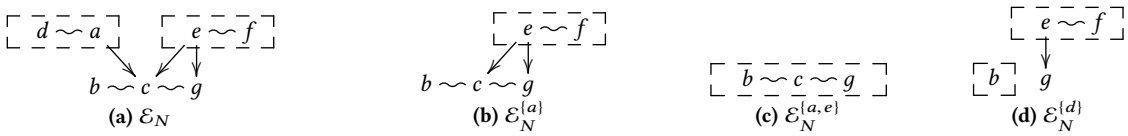


Figure 8. AB's branching cell decomposition (running example)

are given by a formula $\Phi(x)$ of the propositional calculus without negation, where the basic propositions are the transitions of the occurrence net. If we represent such a formula as a sum of products, it corresponds to a set of *collections*, i.e. a set of sets of transitions. Different collections correspond to alternative causal dependencies, while transitions within a collection are all the causes of that alternative and *true* represents the empty collection. Such a formula $\Phi(x)$ represents a monotone boolean function, which expresses, as a function of the occurrences of past transitions, if x has enough causes. It is known that such formulas, based on positive literals only, have a unique DNF (sum of products) form, given by the set of prime implicants. In fact, every prime implicant is also essential [Wegener 1987]. We define $\Phi(x)$ by well-founded recursion:

$$\Phi(x) = \begin{cases} \mathit{true} & \text{if } x \in P \cup \mathbf{P} \wedge {}^\bullet x = \emptyset \\ \bigvee_{t \in {}^\bullet x} (t \wedge \Phi(t)) & \text{if } x \in P \cup \mathbf{P} \wedge {}^\bullet x \neq \emptyset \\ \bigwedge_{s \in {}^\bullet x} \Phi(s) & \text{if } x \in T \end{cases}$$

Ordinary deterministic processes satisfy *complete concurrency*: each process determines a partial ordering of its transitions, such that the executable sequences of transitions are exactly the linearizations of the partial order. More formally, after executing any firing sequence σ of the process, a transition t is enabled if and only if all its predecessors in the partial order (namely its causes) already appear in σ. In the present setting a similar property holds.

Definition 5.2 (Legal firing sequence)**.** A sequence of transitions $t_1; \cdots; t_n$ of a persistent process is *legal* if for all $k \in [1, n]$ we have that $\bigwedge_{i=1}^{k-1} t_i$ implies $\Phi(t_k)$.

It is immediate to notice that if the set of persistent places is empty ($\mathbf{P} = \emptyset$) then the notion of persistent process is the ordinary one, $\Phi(x)$ is just the conjunction of the causes of x and a sequence is legal iff it is a linearization of the process.

Theorem 5.3 (Complete Concurrency)**.** *Let $\sigma = t_1; \cdots; t_n$ with $n \geq 0$ be a, possibly empty, firing sequence of a persistent process, and t a transition not in σ. The following conditions are all equivalent: (i) t is enabled after σ; (ii) there is a collection of causes of t which appears in σ; (iii) $\bigwedge_{i=1}^{n} t_i$ implies $\Phi(t)$.*

Corollary 5.4. *Given a persistent process, a sequence is legal iff it is a firing sequence.*

Example 5.5. Fig. 9 shows a process for the net $(\!|[\![N]\!]|\!)$ of our running example (see N in Fig. 3a and $[\![N]\!]$ in Fig. 7). The process accounts for the firing of the transitions d, f, b in N. Despite they look as concurrent events in N, the persistent place $\mathbf{p}_{\mathrm{t_b}}$ introduces some causal dependencies. In fact, we have: $\Phi(t_d) = \Phi(t_f) = \mathit{true}$, $\Phi(t_3) = t_d$, $\Phi(t_8) = t_f$ and $\Phi(t_b) = (t_3 \wedge t_d) \vee (t_8 \wedge t_f)$, thus t_b can be fired only after either t_d or t_f (or both).

6 Probabilistic Nets

We can now outline our methodology to assign probabilities to the concurrent runs of a Petri net, also in the presence of confusion.

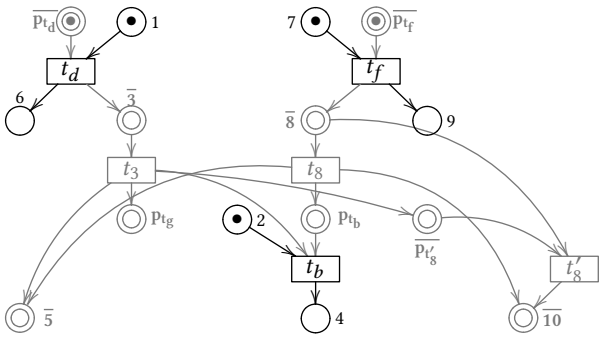


Figure 9. A process for $(\!|[\![N]\!]|\!)$ (running example)

Given a net N, we apply s-cell decomposition from Section 3.1, and then we assign probability distributions to the transactions available in each cell $\mathbb{C}$ (and recursively to the s-cell decomposition of $N_{\mathbb{C}}$). Let $\mathcal{P}_{\mathbb{C}} : \{\theta \mid \theta : \mathbb{C}\} \to [0, 1]$ denote the probability distribution function of the s-cell $\mathbb{C}$ (such that $\sum_{\theta:\mathbb{C}} \mathcal{P}_{\mathbb{C}}(\theta) = 1$). Such probability distributions are defined locally and transferred automatically to the transitions in T_{pos} of the dynamic p-net $[\![N]\!]$ defined in Section 3, in such a way that $\mathcal{P}(t_{\theta,\mathbb{C}}) = \mathcal{P}_{\mathbb{C}}(\theta)$. Each negative transitions in T_{neg} has probability 1 because no choice is associated with it. Since the uniformed net $(\!|[\![N]\!]|\!)$ has the same transitions of $[\![N]\!]$, the probability distribution can be carried over $(\!|[\![N]\!]|\!)$ (thanks to Proposition 2.4).

A simple way to define $\mathcal{P}_{\mathbb{C}}$ is by assigning probability distributions to the arcs leaving the same place of the original net, Then, given a transaction $\theta : \mathbb{C}$, we can set $Q_{\mathbb{C}}(\theta)$ be the product of the probability associated with the arcs of N entering the transitions in θ. Of course, in general it can happen that $\sum_{\theta:\mathbb{C}} Q_{\mathbb{C}}(\theta) < 1$, as not all combinations are feasible. However, it is always possible to normalise the quantities of feasible assignments by setting $\mathcal{P}_{\mathbb{C}}(\theta) = \frac{Q_{\mathbb{C}}(\theta)}{\sum_{\theta':\mathbb{C}} Q_{\mathbb{C}}(\theta')}$ for any transaction $\theta : \mathbb{C}$.

Example 6.1. Suppose that in our running example we assign uniform distributions to all arcs leaving a place. From simple calculation we have $\mathcal{P}_{\mathbb{C}_1}(\theta_a) = \mathcal{P}_{\mathbb{C}_1}(\theta_d) = \frac{1}{2}$ for the first cell, $\mathcal{P}_{\mathbb{C}_2}(\theta_e) = \mathcal{P}_{\mathbb{C}_2}(\theta_f) = \frac{1}{2}$ for the second cell, $\mathcal{P}_{\mathbb{C}_3}(\theta_c) = \mathcal{P}_{\mathbb{C}_3}(\theta_{bg}) = \frac{1}{2}$ for the third cell. The transactions of nested cells are uniquely defined and thus have all probability 1.

Given a firing sequence $t_1; \cdots; t_n$ we can set $\mathcal{P}(t_1; \cdots; t_n) = \prod_{i=1}^{n} \mathcal{P}(t_i)$. Hence firing sequences that differ in the order in which transitions are fired are assigned the same probability. Thanks to Theorem 5.3, we can consider maximal persistent processes and set $\mathcal{P}(O) = \prod_{t \in O} \mathcal{P}(t)$. In fact any maximal firing sequence in O includes all transitions of O. It follows from Theorem 4.3 that any maximal configuration has a corresponding maximal process (and viceversa) and since Abbes and Benveniste proved that the sum of

the probabilities assigned to maximal configurations is 1, the same holds for maximal persistent processes.

Example 6.2. Suppose the distributions assigned in Example 6.1. Then, the persistent process in Fig. 9 has probability: $\mathcal{P}(O) = \mathcal{P}(t_d) \cdot \mathcal{P}(t_f) \cdot \mathcal{P}(t_3) \cdot \mathcal{P}(t_8) \cdot \mathcal{P}(t_b) \cdot \mathcal{P}(t'_8) = \frac{1}{2} \cdot \frac{1}{2} \cdot 1 \cdot 1 \cdot 1 \cdot 1 = \frac{1}{4}$.

7 Conclusion and Future Work

AB's branching cells are a sort of interpreter (or scheduler) for executing PESs in the presence of confusion. Our main results develop along two orthogonal axis. Firstly, our approach is an innovative construction with the following advantages:

1. Compositionality: s-cells are defined statically and locally, while AB's branching cells are defined dynamically and globally (by executing the whole event structure).
2. Compilation vs interpretation: AB's construction gives an interpreter that rules out some executions of an event structure. We instead compile a net into another one (with persistency) whose execution is driven by ordinary firing rules.
3. Complete concurrency: AB's recursively stopped configurations may include traces that cannot be executed by the interpreter. Differently, our notion of process captures all and only those executable traces of a concurrent computation.
4. Simplicity: s-cells definition in terms of a closure relation takes a couple of lines (see Definition 3.1), while AB's branching cell definition is more involved.
5. Full matching: we define a behavioural correspondence that relates AB's maximal configurations with our maximal deterministic processes, preserving their probability assignment.

Secondly, we provide the following fully original perspectives:

1. Confusion removal: our target model is confusion-free.
2. Locally executable model: probabilistic choices are confined to transitions with the same pre-set, and hence can be resolved locally and concurrently. Besides, our target model relies on ordinary firing rules (with persistent places).
3. Processes: We define a novel notion of process for nets with persistency that conservatively extends the ordinary notion of process and captures the right amount of concurrency.
4. Goal satisfaction: our construction meets all requirements in the list of desiderata.

Our construction is potentially complex: given a s-cell $\mathbb{C}$ we recursively consider the nested s-cells in $N_{\mathbb{C}} \ominus p$, for any initial place $p \in N_{\mathbb{C}}$. In the worst case, the number of nested s-cells can be exponential on the number of their initial places. However s-cells are typically much smaller than the whole net and it can be the case that the size of all s-cells is bound by some fixed k. In this case, the number of s-cells in our construction can still become exponential on the constant k, but linear w.r.t. the number of places of the net.

A limitation of our approach is that it applies to finite occurrence nets only (or, equivalently, to finite PESs). As a future work, we plan to deal with cycles and unfolding semantics. This requires some efforts and we conjecture it is feasible only if the net is safe and its behaviour has some regularity: the same s-cell can be executed several times in a computation but every instance is restarted without tokens left from previous rounds. The causal AND/OR-dependencies share some similarities also with the work on connectors and Petri nets with boundaries [Bruni et al. 2013] that we would like to formalize. We also want to investigate the connection between our s-cell structure and Bayesian networks, so to make forward and backward reasoning techniques available in our setting.

Acknowledgments

The idea of structural cells emerged, with a different goal, in collaboration with Lorenzo Galeotti after his MSc thesis. We thank Glynn Winskel, Holger Hermanns, Joost-Pieter Katoen for giving us insightful references. The second author has been partially supported by CONICET grant PIP 11220130100148CO. The third author carried on part of the work while attending a Program on Logical Structures in Computation at Simons Institute, Berkeley, 2016.

References

Samy Abbes and Albert Benveniste. 2005. Branching cells as local states for event structures and nets: Probabilistic applications. In *FOSSACS'05 (Lect. Notes in Comp. Sci.)*, Vol. 3441. Springer, 95–109.

Samy Abbes and Albert Benveniste. 2006. True-concurrency probabilistic models: Branching cells and distributed probabilities for event structures. *Inf. Comput.* 204, 2 (2006), 231–274.

Samy Abbes and Albert Benveniste. 2008. True-concurrency probabilistic models: Markov nets and a law of large numbers. *Theor. Comput. Sci.* 390, 2-3 (2008), 129–170.

Andrea Asperti and Nadia Busi. 2009. Mobile Petri nets. *Mathematical Structures in Computer Science* 19, 6 (2009), 1265–1278.

Anne Bouillard, Stefan Haar, and Sidney Rosario. 2009. Critical paths in the partial order unfolding of a stochastic petri net. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 43–57.

Roberto Bruni, Hernán C. Melgratti, Ugo Montanari, and Pawel Sobocinski. 2013. Connector algebras for C/E and P/T nets' interactions. *Logical Methods in Computer Science* 9, 3 (2013).

Federico Crazzolara and Glynn Winskel. 2005. Petri nets with persistence. *Electr. Notes Theor. Comput. Sci.* 121 (2005), 143–155.

Joanne Bechta Dugan, Kishor S. Trivedi, Robert Geist, and Victor F. Nicola. 1984. Extended stochastic Petri nets: Applications and analysis. In *Performance'84*. North-Holland, 507–519.

Christian Eisentraut, Holger Hermanns, Joost-Pieter Katoen, and Lijun Zhang. 2013. A semantics for every GSPN. In *Petri Nets 2013 (Lect. Notes in Comp. Sci.)*, Vol. 7927. Springer, 90–109.

Ursula Goltz and Wolfgang Reisig. 1983. The non-sequential behavior of Petri nets. *Information and Control* 57, 2/3 (1983), 125–147.

Stefan Haar. 2002. Probabilistic cluster unfoldings. *Fundamenta Informaticae* 53, 3-4 (2002), 281–314.

Joost-Pieter Katoen and Doron A. Peled. 2013. Taming confusion for modeling and implementing probabilistic concurrent systems. In *ESOP'13 (Lect. Notes in Comp. Sci.)*, Vol. 7792. Springer, 411–430.

Joost-Pieter Katoen, Rom Langerak, and Diego Latella. 1993. Modeling systems by probabilistic process algebra: An event structures approach. (1993).

Manfred Kudlek. 2005. Probability in Petri nets. *Fundamenta Informaticae* 67, 1-3 (2005), 121–130.

Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. 1984. A class of Generalized Stochastic Petri Nets for the performance evaluation of Multiprocessor Systems. *ACM Trans. Comput. Syst.* 2, 2 (1984), 93–122.

Michael K. Molloy. 1985. Discrete time stochastic Petri nets. *IEEE Trans. Softw. Eng.* 11, 4 (April 1985), 417–423.

Ugo Montanari and Francesca Rossi. 1995. Contextual nets. *Acta Inf.* 32, 6 (1995), 545–596.

Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. 1981. Petri nets, event structures and domains, part I. *Theor. Comput. Sci.* 13 (1981), 85–108.

C.A. Petri. 1962. *Kommunikation mit Automaten*. Ph.D. Dissertation. Institut für Instrumentelle Mathematik, Bonn.

Wolfgang Reisig. 2013. *Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies*. Springer.

Grzegorz Rozenberg and Joost Engelfriet. 1998. Elementary net systems. In *Advances in Petri Nets 1996, Part I (Lect. Notes in Comp. Sci.)*, Vol. 1491. Springer, 12–121.

Einar Smith. 1996. On the border of causality: Contact and confusion. *Theor. Comput. Sci.* 153, 1&2 (1996), 245–270.

Rob J. van Glabbeek, Ursula Goltz, and Jens-Wolfhard Schicke-Uffmann. 2013. On characterising distributability. *Logical Methods in Computer Science* 9, 3 (2013).

Daniele Varacca, Hagen Völzer, and Glynn Winskel. 2006. Probabilistic event structures and domains. *Theor. Comput. Sci.* 358, 2-3 (2006), 173–199.

Ingo Wegener. 1987. *The complexity of Boolean functions*. John Wiley & Sons, Inc., New York, NY, USA.

Glynn Winskel. 1987. Event structures. In *Advances in Petri Nets 1986, Part II (Lect. Notes in Comp. Sci.)*, Vol. 255. Springer, 325–392.