Contextual Merged Processes*

César Rodríguez[1], Stefan Schwoon[1], and Victor Khomenko[2]

[1] LSV, ENS Cachan & CNRS, INRIA Saclay, France
[2] School of Computing Science, Newcastle University, U.K.

Abstract. We integrate two compact data structures for representing state spaces of Petri nets: merged processes and contextual prefixes. The resulting data structure, called *contextual merged processes (CMP),* combines the advantages of the original ones and copes with several important sources of state space explosion: concurrency, sequences of choices, and concurrent read accesses to shared resources. In particular, we demonstrate on a number of benchmarks that CMPs are more compact than either of the original data structures. Moreover, we sketch a polynomial (in the CMP size) encoding into SAT of the model-checking problem for reachability properties.

1 Introduction

Model checking of concurrent systems is an important and practical way of ensuring their correctness. However, the main drawback of model checking is that it suffers from the *state-space explosion (SSE)* problem [23]. That is, even a relatively small system specification can (and often does) yield a very large state space. To alleviate SSE, many model-checking techniques use a condensed representation of the full state space of the system. Among them, a prominent technique are McMillan's Petri net unfoldings (see, e.g. [15,6,13]). They rely on the partial-order view of concurrent computation and represent system states implicitly, using an acyclic *unfolding prefix*.

There are several common sources of SSE. One of them is concurrency, and the unfolding techniques were primarily designed for efficient verification of highly concurrent systems. Indeed, complete prefixes are often exponentially smaller than the corresponding reachability graphs because they represent concurrency directly rather than by multidimensional 'diamonds' as it is done in reachability graphs. For example, if the original Petri net consists of 100 transitions that can fire once in parallel, the reachability graph will be a 100-dimensional hypercube with 2^{100} vertices, whereas the complete prefix will be isomorphic to the net itself. However, unfoldings do not cope well with some other important sources of SSE, and in what follows, we consider two such sources.

One important source of SSE are sequences of choices. For example, the smallest complete prefix of the Petri net in Fig. 1 is exponential in its size since no event can be declared a cutoff — intuitively, each reachable marking 'remembers' its past, and so different runs cannot lead to the same marking.

* This research was supported by the EPSRC grant EP/K001698/1 (UNCOVER).

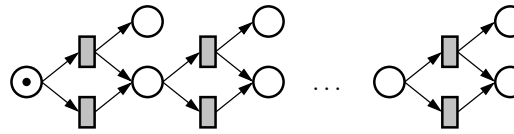

Fig. 1. A Petri net with exponentially large unfolding prefix.

Another important source of SSE are concurrent read accesses, that is, multiple actions requiring non-exclusive access to a shared resource. *Contextual nets (c-nets)* are an extension of Petri nets where such read accesses are modelled by a special type of arcs, called *read arcs* and denoted by lines (in contrast to arrows for the traditional consuming and producing arcs). Read arcs allow a transition to check for the presence of a token without consuming it. As concurrent read access to a shared resource is a natural operation in many concurrent systems, c-nets are often the formalism of choice for a wide variety of applications, e.g. to model concurrent database access [18], concurrent constraint programs [16], priorities [9], and asynchronous circuits [24].

The usual way of modelling c-nets using traditional Petri nets is by replacing read arcs by "consume-reproduce loops": a transition consumes a token from a place and immediately puts a token back, see Fig. 2 (a,b). Unfortunately, this makes the unfolding technique inefficient: concurrent transitions of a c-net reading the same place are sequentialised by this encoding, and thus all their interleavings are represented in the unfolding, see Fig. 3 (b). This problem can be mitigated using the place-replication (PR) encoding proposed in [24], which replicates each place that is read by several transitions so that each of them obtains a "private" copy of the place and accesses it using a consume-reproduce loop, see Fig. 2 (a,c). However, the resulting unfolding may still be large, see Fig. 3 (c). Moreover, the PR encoding can significantly increase the sizes of presets of some transitions, considerably slowing down the unfolding algorithm, because (with some reasonable assumptions) the problem of checking if the currently built part of the prefix can be extended by a new instance of a transition t is NP-complete in the prefix size and $|{}^\bullet t|$ [7, Sect. 4.4].

Recently, techniques addressing these sources of SSE emerged. In [14], a new condensed representation of Petri net behaviour called *merged processes (MPs)* was proposed, which copes not only with concurrency, but also with sequences of choices. Moreover, this representation is sufficiently similar to the traditional unfoldings so that a large body of results developed for unfoldings can be re-used. The main idea behind MPs is to fuse some nodes in the unfolding prefix, and use the resulting net as the basis for verification. For example, the unfolding of the net shown in Fig. 1 will collapse back to the original net after the fusion. It turns out that for a safe Petri net, model checking of a reachability-like property (i.e. the existence of a reachable state satisfying a predicate given by a Boolean expression) can be efficiently performed on its MP, and [14] provides a polynomial reduction of this problem to SAT. Furthermore, an efficient *unravelling algorithm* that builds a complete MP of a given safe PN has been proposed in [11].

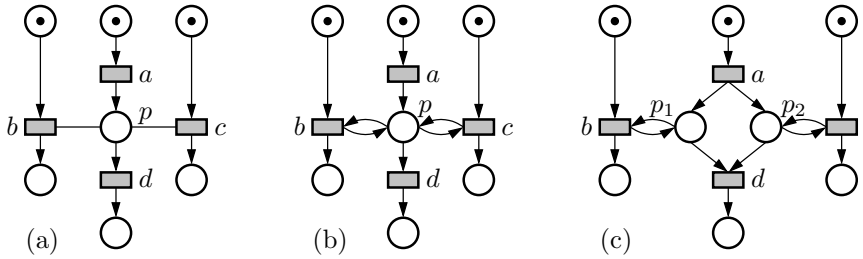

Fig. 2. (a) A c-net; (b) its *plain encoding*; (c) and its *place-replication encoding*.

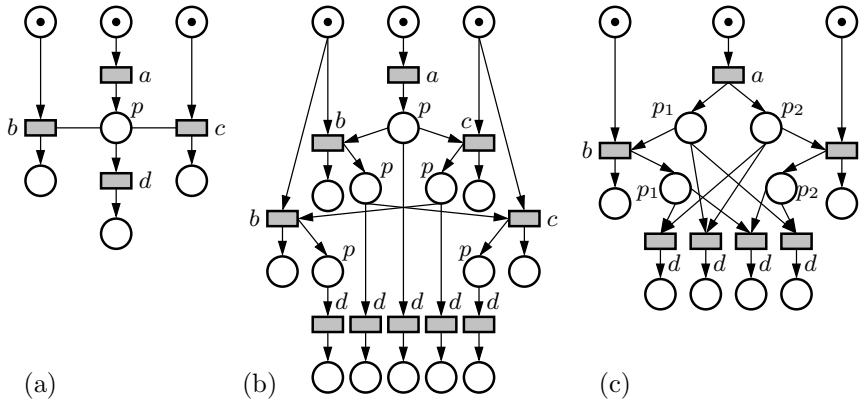

Fig. 3. Unfoldings of (a), (b), and (c) in Fig. 2

The experimental results in [14] indicate that this method is quite practical. Unfortunately, MPs do not cope well with read arcs, as illustrated in Fig. 4.

An extension of the unfolding technique to c-nets was proposed in [3,24], and a practical unfolding algorithm and SAT-based model checking for reachability-like properties have been developed in [1,20]. The idea is to allow read arcs also in the unfolding, which allows for significant compression in some cases — see Fig. 3(a). The experimental results in [1,20] demonstrate that the performance of this method is comparable to the traditional unfoldings when c-nets have no read arcs (i.e. can be directly interpreted as Petri nets), and can be much better (in terms of both the runtime and the size of the generated prefix) than traditional unfolding of plain and PR encodings of c-nets with many read arcs. Unfortunately, this method does not cope with SSE resulting from sequences of choices, e.g. it does not offer any improvement for the Petri net in Fig. 1, as it contains no read arcs.

In this paper we observe that the described techniques for compressing the unfolding prefix are in fact orthogonal, and can be combined into one that copes with all the mentioned sources of SSE, viz. concurrency, sequences of choices and concurrent read accesses to a shared resource. Moreover, there are striking

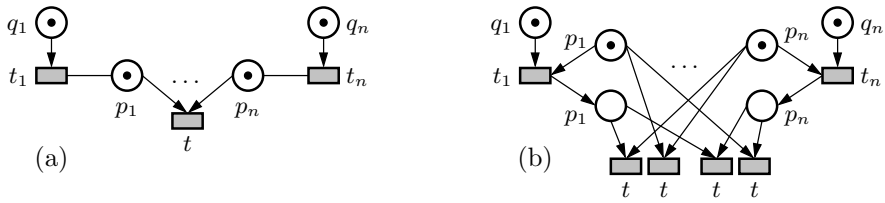

Fig. 4. A c-net (a) whose contextual unfolding is isomorphic to the c-net itself, but whose plain encoding into a Petri net has exponentially large merged process, since no place instances in its unfolding (b) can be merged, and so there are 2^n mp-events corresponding to transition t. (For this c-net the PR encoding coincides with the plain one, and so has the same unfolding and MP.)

similarities between the main complications that had to be overcome in the theories of MPs and c-net unfoldings: events have multiple local configurations (which causes difficulties in detection of cutoff events), and certain cycles (in the flow relation in case of MPs and in the asymmetric conflict relation in case of c-net unfoldings) have to be prohibited in valid configurations. Hence, the combination of the two techniques is not only possible, but also is very natural.

The paper is organised as follows. In Section 2 we provide the necessary definitions related to c-nets and unfoldings. Section 3 — the main contribution of this paper — introduces the notion of a *contextual merged process (CMP)* and provides results to characterise the configurations of CMPs of safe c-nets. We use these results in Section 4 to discuss the construction and SAT-based model checking of CMPs. In Section 5 we experimentally evaluate the proposed approach on a number of benchmark examples. In Section 6 we conclude and outline the directions for future research.

A longer version of this paper, including proofs, is available at [22].

2 Basic Notions

In this section, we set our basic definitions and recall previous results (see [2,21]).

A *multiset* over a set S is a function $M\colon S \to \mathbb{N}$. The *support* of M is the set $\bar{M} := \{x \in S \mid M(x) > 0\}$ of elements in S occurring at least once in M. We write $x \in M$ if x is in the support of M. We say that M is finite iff its support is. Given multisets M and N over S, their sum and difference are $(M + N)(x) := M(x) + N(x)$ and $(M - N)(x) := \max(0, M(x) - N(x))$. We write $M \leq N$ iff $M(x) \leq N(x)$ for all $x \in S$. Any function $f\colon S \to T$ can be lifted to multisets by letting $f(M)(x) := \sum_{y \in f^{-1}(x)} M(y)$; note that this sum is well-defined iff finitely many of its summands are non-zero, which is always the case if, for instance, M has a finite support. Any set can be interpreted as a multiset in the natural way.

A *contextual net* (c-net) is a tuple $N = \langle P, T, F, C, m_0 \rangle$, where P and T are disjoint sets of *places* and *transitions*, $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*,

$C \subseteq P \times T$ is the *context relation*, and the *initial marking* m_0 is a multiset over P. A pair $(p,t) \in C$ is called *read arc*. A *Petri net* is a c-net without read arcs. N is called *finite* if P and T are finite sets. Places and transitions together are called *nodes*. Fig. 2 (a) depicts a c-net, where read arcs are drawn as undirected lines, e.g. between p and c.

For $x \in P \cup T$, ${}^\bullet x := \{\, y \in P \cup T \mid (y,x) \in F \,\}$ is the *preset* of x, $x^\bullet := \{\, y \in P \cup T \mid (x,y) \in F \,\}$ is the *postset* of x, and $\underline{x} := \{\, y \in P \cup T \mid (y,x) \in C \cup C^{-1} \,\}$ is the *context* of x. We assume that for each node $x \in P \cup T$ the sets ${}^\bullet x$, $x^\bullet$, and $\underline{x}$ are pairwise disjoint.

A *marking* of N is a multiset m over P. A transition t is *enabled* at m if $m(p) \geq 1$ for all $p \in \underline{t} \cup {}^\bullet t$. Such t can *fire*, leading to the well-defined marking $m' := m - {}^\bullet t + t^\bullet$. The tuple $\langle m, t, m' \rangle$ is called a *step*. A marking m is *reachable* if it can be obtained by a finite sequence of firings starting at m_0. N is *k-bounded* if $m(p) \leq k$ for all reachable m and all $p \in P$, and *safe* if it is 1-bounded. For safe nets, we treat markings as sets of places.

Two distinct transitions t and t' are in *symmetric conflict*, denoted $t \# t'$, if ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$, and in *asymmetric conflict*, written $t \nearrow t'$, if (i) $t^\bullet \cap ({}^\bullet t' \cup \underline{t'}) \neq \emptyset$, or (ii) $\underline{t} \cap {}^\bullet t' \neq \emptyset$, or (iii) $t \# t'$. Intuitively, when $t \nearrow t'$, then if both t, t' fire in a run, t fires before t'. Note that t and t' may not fire together in any run, e.g. if $t \# t'$, where we have $t \nearrow t'$ *and* $t' \nearrow t$ — corresponding to the intuition that t has to fire before t' and vice versa. In Fig. 6 (b) we have $e_3 \nearrow e_5$ due to (i); in Fig. 5 we have $e_1 \nearrow e_2$ due to (ii). For a set of transitions $X \subseteq T$, we write $\nearrow_X$ to denote the relation $\nearrow \cap\, (X \times X)$.

Let $N' = \langle P', T', F', C', m'_0 \rangle$ be a c-net. A *homomorphism* [24] from N to N' is a function $h\colon P \cup T \to P' \cup T'$ satisfying: $h(P) \subseteq P'$, $h(T) \subseteq T'$, $h(m_0) = m'_0$, and h restricted to ${}^\bullet t$, $t^\bullet$, $\underline{t}$ for all $t \in T$ is a bijection to ${}^\bullet h(t)$, $h(t)^\bullet$ and $\underline{h(t)}$, respectively. Such a homomorphism is a specialisation of Definition 4.20 in [3].

For two nodes x and y we write $x <_i y$ if either $(x,y) \in F$ or $x, y \in T$ and $x^\bullet \cap \underline{y} \neq \emptyset$. We write $<$ for the transitive closure of $<_i$, and $\leq$ for the reflexive closure of $<$. For a node x, we define its set of *causes* as $[x] := \{\, t \in T \mid t \leq x \,\}$. A set $X \subseteq T$ is *causally closed* if $[t] \subseteq X$ for all $t \in X$.

An *occurrence net* is a c-net $O = \langle B, E, G, D, \tilde{m}_0 \rangle$ if (i) O is safe and for any $b \in B$, we have $|{}^\bullet b| \leq 1$; (ii) $<$ is a strict partial order for O; (iii) for all $e \in E$, $[e]$ is finite and $\nearrow_{[e]}$ acyclic; (iv) $\tilde{m}_0 = \{\, b \in B \mid {}^\bullet b = \emptyset \,\}$. As per tradition, we call the elements of B *conditions*, and those of E *events*. A *configuration* of O is a finite, causally closed set of events $\mathcal{C}$ such that $\nearrow_{\mathcal{C}}$ is acyclic; $\mathit{Conf}(O)$ denotes the set of all configurations. For a configuration $\mathcal{C}$, let $\mathit{cut}(\mathcal{C}) := (\tilde{m}_0 \cup \mathcal{C}^\bullet) \setminus {}^\bullet\mathcal{C}$. A *prefix* of O is a c-net $\mathcal{P} = \langle B', E', G', D', \tilde{m}_0 \rangle$ such that $E' \subseteq E$ is causally closed, $B' = \tilde{m}_0 \cup E'^\bullet$, and G' and D' are the restrictions of G and D to $B' \cup E'$; in such a case we write $\mathcal{P} \sqsubseteq O$.

Fig. 5 shows an occurrence net illustrating why it is necessary to restrict configurations to sets without cycles in $\nearrow$. There are three events, and each pair of them can fire, but not all three. Indeed, $e_1 \nearrow e_2 \nearrow e_3 \nearrow e_1$ is a cycle of asymmetric conflicts.

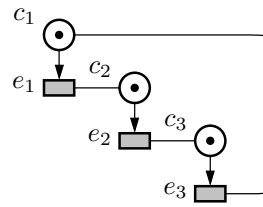

Fig. 5. An occurrence net illustrating circular asymmetric conflict.

A *branching process* of N is a pair $\mathcal{P} = \langle O, h \rangle$, where O is an occurrence net and h is a homomorphism from O to N with the property that $h(e) = h(e') \wedge {}^\bullet e = {}^\bullet e' \wedge \underline{e} = \underline{e'}$ implies $e = e'$ for all events $e, e' \in O$. For every N, there is a unique (up to isomorphism) maximal (wrt. $\sqsubseteq$) branching process $\mathcal{U}_N = \langle U, h' \rangle$ that we call the *unfolding* of N [2]. Thus, any branching processes $\langle O, h \rangle$ is characterised by a prefix O of U and the restriction h of h' to the elements of O. For convenience, we shall often equate a branching process with its underlying net and call it an *unfolding prefix*. As usual, for $\mathcal{C} \in \mathit{Conf}(\mathcal{U}_N)$, we define $\mathit{mark}(\mathcal{C}) := h(\mathit{cut}(\mathcal{C}))$.

An unfolding prefix $\mathcal{P}$ is called *marking-complete* if for any marking m reachable in N there exists a marking $\tilde{m}$ reachable in $\mathcal{P}$ with $h(\tilde{m}) = m$. For example, $\mathcal{U}_N$ is marking-complete but in general infinite. For bounded N, it is however possible to compute a finite marking-complete prefix $\mathcal{P}_N$ [2,21].

The key notion in computing marking-complete prefixes is a *history*. Given a configuration $\mathcal{C} \in \mathit{Conf}(U)$ and some event $e \in \mathcal{C}$, the *history of e in $\mathcal{C}$* is defined as $\mathcal{C}[\![e]\!] := \{\, e' \in \mathcal{C} \mid e' \nearrow_{\mathcal{C}}^* e \,\}$. For $e \in E$, $\mathit{Hist}(e) := \{\, \mathcal{C}[\![e]\!] \mid \mathcal{C} \in \mathit{Conf}(U) \,\}$ is the set of *all histories* of e. The construction of a complete prefix discovers events that do not contribute to reaching new markings of N in the prefix: an event e is declared *cutoff* if for every history H of e there exists a configuration $\mathcal{C}$ in $\mathcal{P}$ such that $\mathit{mark}(\mathcal{C}) = \mathit{mark}(H)$ and $\mathcal{C} \prec H$, where $\prec$ is a so-called *adequate order* on configurations.[3] The construction then excludes events that are causal successors of e, thereby ensuring the finiteness of $\mathcal{P}$ while guaranteeing its marking-completeness: for every reachable marking m of N there is a configuration $\mathcal{C}$ of $\mathcal{P}$ such that $\mathit{mark}(\mathcal{C}) = m$ and $\mathcal{C}$ does not include any cutoffs.

3 Contextual Merged Processes

In this section, we introduce the notion of *contextual merged processes* (CMP) and discuss some of their properties. These results generalise those of [14], in particular it turns out that the notions of mp-configuration, defined in [14] for Petri

[3] Actually, [2] defines pairs $\langle e, H \rangle$ as cutoffs; above, we chose an equivalent presentation that will be more convenient for defining CMPs. Also, only histories are considered for $\mathcal{C}$ in [2]; we come back to this point in Section 4.

nets, and the notion of a c-net configuration from [2], both of which introduce acyclicity constraints, can be seamlessly integrated into a common framework.

We first show (see [22] for the proofs of all results) that asymmetric conflict, causality, and steps are, among other notions, preserved by homomorphisms.

Lemma 1. *Let N and N' be c-nets, and h be a homomorphism from N to N'. If $\langle m, t, \widehat{m}\rangle$ is a step of N and $h(m)$ is well-defined,*[4] *then $\langle h(m), h(t), h(\widehat{m})\rangle$ is a step of N'. Furthermore, for any nodes x, y and transitions t, u of N, $x < y$ implies $h(x) < h(y)$ and $t \nearrow u$ implies either $h(t) \nearrow h(u)$ or $h(t) = h(u)$.*

As usual, homomorphisms preserve runs and reachable markings: if σ is a run of N that reaches m, then $h(\sigma)$ is a run of N' that reaches $h(m)$, because $h(m_0) = m'_0$ is a well-defined marking and due to Lemma 1.

The first step to define CMPs is the notion of occurrence depth.

Definition 1 (occurrence depth). *Let x be a node of a branching process $\langle O, h\rangle$. The* occurrence depth *of x, denoted* $\mathrm{od}(x)$*, is the maximum number of $h(x)$-labelled nodes in any path in the directed graph $(\tilde{m}_0 \cup [x] \cup [x]^\bullet, <_i)$ starting at any initial condition and ending in x.*

Recall that the cone $[x]$ is finite and $<_i$ is a partial order, so there is only a finite number of paths to evaluate, and the definition is well-given.

A CMP is obtained from a branching process in two steps. First, all conditions that have the same label and occurrence depth are fused together (their initial markings are totalled); then all events that have the same label and environment (after fusing conditions) are merged. Conditions in the initial marking will have, by definition, occurrence depth 1. If n of them share the same label, they will be fused together, and the resulting condition will be initially marked with n tokens. This is formalised as follows:

Definition 2 (contextual merged process). *Let $N = \langle P, T, F, C, m_0\rangle$ be a c-net and $\mathcal{P} = \langle\langle B, E, G, D, \tilde{m}_0\rangle, h\rangle$ be a branching process of N. Define a net $\mathcal{Q} = \langle \widehat{B}, \widehat{E}, \widehat{G}, \widehat{D}, \widehat{m}_0\rangle$, where $\widehat{B} \subseteq P \times \mathbb{N}$, $\widehat{E} \subseteq T \times 2^{\widehat{B}} \times 2^{\widehat{B}} \times 2^{\widehat{B}}$, and a homomorphism $\hbar$ from $\mathcal{P}$ to $\mathcal{Q}$ as follows:*

- *for $b \in B$, $\hbar(b) := \langle h(b), \mathrm{od}(b)\rangle$; set $\widehat{B} := \hbar(B)$;*
- *for $e \in E$, $\hbar(e) := \langle h(e), \hbar({}^\bullet e), \hbar(\underline{e}), \hbar(e^\bullet)\rangle$; set $\widehat{E} := \hbar(E)$;*
- *$\widehat{G}, \widehat{D}$ are such that for every $\widehat{e} = \langle t, Pre, Cont, Post\rangle \in \widehat{E}$ we have ${}^\bullet\widehat{e} := Pre$, $\underline{\widehat{e}} := Cont$, $\widehat{e}^\bullet := Post$;*
- *$\widehat{m}_0(\langle p, d\rangle) := |\tilde{m}_0 \cap \{\, b \in B \colon h(b) = p,\ \mathrm{od}(b) = d \,\}|$.*

Moreover, let $\widehat{h}$ be the homomorphism from $\mathcal{Q}$ to N given by projecting the nodes of $\mathcal{Q}$ to their first components. We call $\mathfrak{Merge}(\mathcal{P}) := \langle \mathcal{Q}, \widehat{h}\rangle$ the merged process *of $\mathcal{P}$. The merged process $\mathcal{M}_N := \mathfrak{Merge}(\mathcal{U}_N)$ of the unfolding of N is called the* unravelling *of N.*

[4] That is, $h(m)$ is a well-defined multiset.

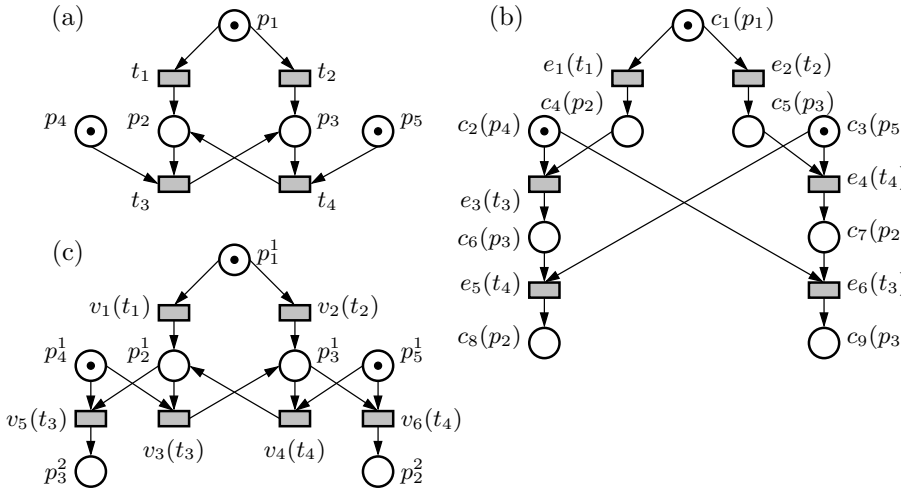

Fig. 6. (a) A net; (b) its unfolding; (c) its unravelling.

Fig. 6 shows a 1-safe net (taken from [14]), its unfolding, and its unravelling. For the rest of this section, let $N = \langle P, T, F, C, m_0 \rangle$ be a bounded c-net, $\mathcal{U}_N$ be its unfolding, $\mathcal{P}_N = \langle \langle B, E, G, D, \tilde{m}_0 \rangle, h \rangle$ be a branching process of N, and $\mathcal{Q}_N = \langle \langle \widehat{E}, \widehat{B}, \widehat{G}, \widehat{D}, \widehat{m}_0 \rangle, \widehat{h} \rangle$ be the corresponding merged process, i.e. $\mathfrak{Merge}(\mathcal{P}_N)$. The places of $\mathcal{Q}_N$ are called *mp-conditions* and its transitions *mp-events*. We shall write p^d for an mp-condition $\langle p, d \rangle$. Note that $\widehat{m}_0(p^d)$ equals $m_0(p)$ if $d = 1$ and is 0 otherwise. An mp-event $\widehat{e}$ is an *mp-cutoff* if all events in $\hbar^{-1}(\widehat{e})$ are cutoffs. We denote these mp-events by $\widehat{E}_{\text{cut}}$.

We call a run $t_1 t_2 \ldots$ of a c-net *repetition-free* if no transition occurs more than once in it. Some properties of contextual merged processes follow.

Remark 1. The following properties hold for CMPs or c-net unfoldings:

1. In general, $\mathcal{M}_N$ is not acyclic; see Fig. 6 (c).
2. There can be mp-events consuming conditions in the postset of an mp-cutoff.
3. There is at most one mp-condition p^k resulting from fusing occurrences of place p at depth $k \geq 1$.
4. For two mp-conditions p^k and p^{k+1}, there is a directed path in the $<_i$ relation from the former to the latter.
5. Two different conditions c_1 and c_2 having the same label and occurrence depth are not causally related. Hence, if the original c-net is safe, then $\nearrow_{[c_1] \cup [c_2]}$ contains a cycle.
6. $h = \widehat{h} \circ \hbar$.
7. $\hbar$ and $\widehat{h}$ are homomorphisms.
8. A sequence of transitions σ is a run of N iff there exists a run $\widehat{\sigma}$ of $\mathcal{M}_N$ such that $\sigma = \widehat{h}(\widehat{\sigma})$.

Additionally, if N is safe, we have:

9. $\hbar$ is injective when restricted to the events of a configuration.
10. Property 8 is true if we additionally require $\widehat{\sigma}$ to be repetition-free.

Note that Property 9 is still true when $\hbar$ is restricted to the elements of $\tilde{m}_0 \cup \mathcal{C} \cup \mathcal{C}^\bullet$. Indeed, $\hbar$ is bijective when restricted to $\tilde{m}_0$, because $\widehat{m}_0$ is safe, and two conditions $c, c' \in \mathcal{C}^\bullet$ cannot be merged because $\nearrow_{[c] \cup [c']}$ would have cycles and $[c] \cup [c'] \subseteq \mathcal{C}$.

Definition 3 (mp-configuration). *A multiset of mp-events $\widehat{\mathcal{C}}$ is an* mp-configuration *of $\mathcal{Q}_N$ if there exists a configuration $\mathcal{C}$ of $\mathcal{U}_N$ verifying $\hbar(\mathcal{C}) = \widehat{\mathcal{C}}$.*

As it is the case for configurations of branching processes, any mp-configuration of a merged process represents a (concurrent) run of its mp-events, i.e. there exists at least one linear ordering of the mp-events of $\widehat{\mathcal{C}}$ that is a run of the merged process. This is because the same is true for configurations of the associated branching process and because $\hbar$ is a homomorphism.

Every finite firing sequence of $\mathcal{U}_N$ consists of a set of events that form a configuration $\mathcal{C}$, which, due to Definition 3, corresponds to an mp-configuration $\widehat{\mathcal{C}}$ of $\mathcal{M}_N$. However, the inverse statement is not true: a firing sequence of $\mathcal{M}_N$ may consist of a multiset of events X that is not an mp-configuration since no $\mathcal{C} \in \mathit{Conf}(\mathcal{U}_N)$ satisfies $\hbar(\mathcal{C}) = X$. This already holds for nets without read arcs, as the example in Fig. 6 shows: $v_1 v_5$ is a valid firing sequence of $\mathcal{M}_N$ corresponding to events e_1 and e_6 of $\mathcal{U}_N$ (i.e. $\hbar(e_1) = v_1$ and $\hbar(e_6) = v_5$) which do not form a configuration. However, $\widehat{h}$ applied to $v_1 v_5$ still gives a valid firing sequence $t_1 t_3$ of N thanks to Remark 1 (8). Below we formalise these observations.

Definition 4 (marking-complete CMP). *Let X be a finite multiset of mp-events. The* cut *and* marking *of X are respectively defined as the multisets*

$$\mathit{cut}(X) := (\widehat{m}_0 + X^\bullet) - {}^\bullet X \qquad \text{and} \qquad \mathit{mark}(X) := \widehat{h}(\mathit{cut}(X)).$$

We call $\mathcal{Q}_N$ marking-complete *if for each reachable marking m of N there exists a cutoff-free mp-configuration $\widehat{\mathcal{C}}$ in $\mathcal{Q}_N$ satisfying $\mathit{mark}(\widehat{\mathcal{C}}) = m$.*

The intuition behind these definitions is as follows. If X is the multiset of mp-events associated to a finite run (i.e. the multiset M such that $M(\widehat{e}) = n$ if $\widehat{e}$ fires n times) then $\mathit{cut}(X)$ is the marking reached by this run in the CMP, and $\mathit{mark}(X)$ is the $\widehat{h}$-image of $\mathit{cut}(X)$, i.e. the corresponding marking of N.

Observe that in the definition of a marking-complete CMP, one could ask for a finite run (rather than a configuration) that reaches a marking m. The resulting definition would be equivalent, but we preferred the current variant because it (i) mimics the analogous definition for unfoldings and (ii) avoids some unpleasant properties of runs: e.g. finite CMPs can have infinite runs and therefore infinitely many finite runs, which is impossible for configurations.

We now focus on the practically relevant class of safe c-nets. Here, the mapping $\hbar$ lifted to configurations establishes an injective correspondence between the configurations of the unfolding and the mp-configurations of the unravelling.

For each mp-configuration $\widehat{\mathcal{C}}$ there exists a *unique* configuration $\mathcal{C}$ such that $\widehat{\mathcal{C}} = \hbar(\mathcal{C})$.

We give, for safe nets, characterisations of sets of mp-events that correspond to reachable markings of N (Proposition 1) and to configurations of $\mathcal{U}_N$ (Proposition 2). They serve to aid CMP-based model-checking, as well as the direct construction of CMPs, see Section 4. We note that the problem of generalising these approaches to bounded, but not safe, nets is still open even for merged processes without read arcs [14].

Proposition 1. *Let $\mathcal{Q}_N$ be a marking-complete CMP of a safe c-net N. Then a marking m is reachable in N iff there exists a cutoff-free set X of mp-events of $\mathcal{Q}_N$ satisfying:*

1. $\forall \widehat{e} \in X : \forall \widehat{c} \in {}^\bullet\widehat{e} \cup \underline{\widehat{e}} : (\widehat{c} \in \widehat{m}_0 \vee \exists \widehat{e}' \in {}^\bullet\widehat{c} : \widehat{e}' \in X)$, *and*
2. $\nearrow_X$ *is acyclic, and*
3. $m = mark(X)$.

Note that the conditions in Proposition 1 do not ensure that X is an mp-configuration; however, they do guarantee that X corresponds to a repetition-free run of $\mathcal{Q}_N$, and thus are sufficient to check reachability (see the comment before Definition 4 for an example). Finally, observe that not every repetition-free run satisfies the first two conditions of Proposition 1: $v_1v_3v_4$ is a repetition-free run of Fig. 6 but $\{v_1, v_3, v_4\}$ violates the second condition. This means that Proposition 1 characterizes a strict subset of repetition-free runs that are enough for representing *all* reachable markings of N.

Proposition 2. *If N is safe, a set of mp-events $\widehat{\mathcal{C}}$ is an mp-configuration of $\mathcal{Q}_N$ iff it satisfies the following conditions:*

1. $\forall \widehat{e} \in \widehat{\mathcal{C}} : \forall \widehat{c} \in {}^\bullet\widehat{e} \cup \underline{\widehat{e}} : (\widehat{c} \in \widehat{m}_0 \vee \exists \widehat{e}' \in {}^\bullet\widehat{c} : \widehat{e}' \in \widehat{\mathcal{C}})$, *and*
2. $\nearrow_{\widehat{\mathcal{C}}}$ *is acyclic, and*
3. *for $k \geq 1$, $p^{k+1} \in \widehat{\mathcal{C}}^\bullet$ implies $p^k \in \widehat{m}_0 \cup \widehat{\mathcal{C}}^\bullet$ and there exists a path in the directed graph $(\widehat{m}_0 \cup \widehat{\mathcal{C}} \cup \widehat{\mathcal{C}}^\bullet, <_i)$ between p^k and p^{k+1}.*

A key detail in both results is that acyclicity of $\nearrow$ prohibits, at the same time, asymmetric conflicts inherent to c-net unfoldings (Fig. 5) and cycles in the flow relation introduced by merging (Fig. 6 (c)).

4 Computing and Analysing Complete CMPs

In this section, we discuss various algorithmic aspects of CMPs, in particular how to construct a complete CMPs from a given *safe* Petri net N, and how to use the resulting CMP to check properties of N.

4.1 CMP Construction

Recall that a marking-complete CMP is one in which every reachable marking m of N is the image (through $\widehat{h}$) of the cut of some cutoff-free mp-configuration. We wish to construct such a CMP in order to analyse properties of N such as reachability or deadlock.

Indirect Methods. It follows from Section 3 that one can achieve this goal by (i) constructing a marking-complete unfolding prefix $\mathcal{P}$ and (ii) applying the construction from Definition 2 to $\mathcal{P}$. Available options for step (i) are:

1. Directly construct $\mathcal{P}$ from N. This approach is implemented in the tool CUNF [19], which is based on the results from [21].
2. Replace all read arcs by consume-produce loops (cf. Fig. 2 (b)) and unfold the resulting Petri net using, e.g., the tool PUNF [12], obtaining some complete prefix $\mathcal{P}'$. We then apply a "folding" operation to $\mathcal{P}'$ in which we repeatedly carry out the following steps: (i) all conditions that were created due to a consume-produce loop are merged and their flow arcs replaced by a read arc; (ii) all events with the same label and the same preset after (i) are merged, and so are their postsets. The resulting c-net prefix $\mathcal{P}$ has the same reachable markings as $\mathcal{P}'$ and is therefore marking-complete. Indeed, applying this operation to the prefix in Fig. 3 (b), which is the unfolding of Fig. 2 (b), would yield the c-net unfolding from Fig. 3 (a).
3. A similar approach as before, but using the place-replication (PR) encoding and adapting the folding operation accordingly (see Fig. 2 (c) and Fig. 3 (c)).

While the first approach is usually more efficient than the others [21], certain aspects of the currently available tool support make options 2 and 3 interesting for the purposes of comparing the resulting CMP sizes. For instance, as pointed out in Footnote 3, CUNF declares a pair $\langle e, H\rangle$, where H is a history of e, a cutoff if it finds another pair $\langle e', H'\rangle$ with $\mathit{mark}(H') = \mathit{mark}(H)$ and $H' \prec H$; this was motivated by the approach from Petri net unfolding [6], where an event is declared cutoff if its local configuration leads to the same marking as the local configuration of another event. However, PUNF implements an approach for Petri nets in which more general configurations are considered for the role of H' [8], leading to smaller unfolding sizes.

Direct Method. Another option is to construct a CMP directly from the c-net N. A similar approach for nets without read arcs was presented in [11]. No such implementation currently exists for CMPs; in the following we describe some key elements that are required for extending [11] to CMPs.

A procedure for direct CMP construction would start with a CMP containing mp-conditions that represent the initial marking of N and extend it one mp-event at a time. To know whether the current CMP $\mathcal{Q}$ can be extended by an mp-event $\widehat{e}$, one has to identify an mp-configuration $\widehat{\mathcal{C}}$ of $\mathcal{Q}$ and check (i) whether $\widehat{\mathcal{C}} \cup \{\widehat{e}\}$ is an mp-configuration of $\mathcal{M}_N$ and (ii) whether $\widehat{e}$ constitutes a cutoff.

Problem (i) can be formulated as a variant of the model-checking algorithm based on Proposition 2 that can be encoded in SAT, see Section 4.2. For (ii), observe that an mp-configuration $\widehat{H}$ corresponds to some history H of an event e of $\mathcal{U}_N$ with $\hbar(e) = \widehat{e}$ iff $\widehat{e}$ is the maximal element of the relation $\nearrow_{\widehat{H}}$. The problem then corresponds to asking whether for all such $\widehat{H}$ there exists another mp-configuration $\widehat{\mathcal{C}}$ such that $\mathit{mark}(\widehat{\mathcal{C}}) = \mathit{mark}(\widehat{H})$ and $\widehat{\mathcal{C}} \prec \widehat{H}$. For $\widehat{\mathcal{C}}, \widehat{H}$ in $\mathcal{Q}$, this problem can be encoded in 2QBF, which is more complicated than SAT but less so than QBF in general, and for which specialised solutions exist [17].

However, as $\mathcal{Q}$ grows, the number of possible candidates for $\widehat{H}$ may increase. In general $\widehat{e}$ cannot be designated a cutoff until the construction has been terminated, instead the possibility of adding $\widehat{e}$ may have to be re-checked periodically.

To summarise, the basic structure of the algorithm from [11] would remain unchanged, however one needs to use the characterisation 2 of mp-configurations rather than the non-contextual one in [11].

4.2 Model Checking CMPs

Let $\mathcal{Q}$ be a CMP. We briefly discuss a possible encoding for runs and mp-configurations of $\mathcal{Q}$ into SAT, using Propositions 1 and 2. Note that [14] discusses the corresponding problems for non-contextual MPs and [20] for contextual unfoldings. Remarkably, both problems require to encode acyclicity for different purposes, which are united into a single acyclicity constraint in our case.

Proposition 1 says that every reachable marking m of N is represented by some $\nearrow$-acyclic run X of $\mathcal{Q}$. Reachability of m reduces, then, to the satisfiability of a SAT formula that has variables c, e, p for mp-conditions, mp-events, and places, respectively, such that e is true iff $\widehat{e} \in X$, c iff $\widehat{c} \in \mathit{cut}(X)$ and p iff $p \in \mathit{mark}(X)$.

Condition 1 of Proposition 1 demands that every event needs a causal predecessor for all non-initial mp-conditions in its preset or context. Condition 3 imposes that the variables for mp-conditions and places be correctly related and that the place variables correspond to m. Both these conditions can easily be encoded in linear size wrt. $|\mathcal{Q}|$. For the acyclicity constraint (Condition 2) there are multiple encodings of polynomial size. We refer the reader to [14, 20], where such encodings are discussed and experimentally evaluated.

Proposition 2, used for constructing CMPs, differs from Proposition 1 in having a more restrictive third condition. This constraint and its encoding is very similar to the "no-gap" constraint from [14] to which we refer the reader for details.

5 Experiments and Case Studies

In this section, we experimentally[5] compare the sizes of CMPs, MPs, and unfoldings for a number of families of c-nets. In Section 5.1, we discuss an artificial family of examples that allows one to study the effects of read arcs and choice for the various methods in isolation. Section 5.2 presents a case study on Dijkstra's mutual exclusion protocol. Finally, Section 5.3 shows how our methods behave on assorted practical benchmarks.

[5] All the benchmarks and tools referenced in this section are publicly available from http://www.lsv.ens-cachan.fr/~rodriguez/experiments/pn2013/.

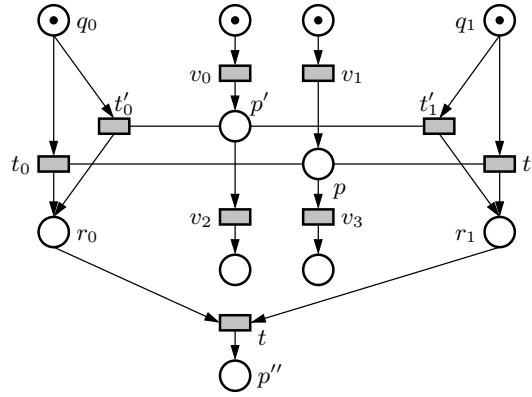

Fig. 7. The c-net 2-Gen.

5.1 Interplay Between Read-Arcs and Choice

We study a family of c-net examples called n-Gen, shown for $n = 2$ in Fig. 7. The net represents n processes that concurrently generate resources r_i. Once all r_is are produced, an action t consumes them all. Resource r_i can be produced if one of two conditions is fulfilled, symbolised by transitions t_i or t'_i. Thus, t_i, t'_i share context with transitions t_j and t'_j, respectively, whenever $j \neq i$.

For some $n \geq 1$, let N_c be the c-net n-Gen, N_p its plain encoding, and N_r its PR encoding. The unfoldings of the three nets and the MPs of N_p and N_r blow up due to at least one of the following reasons, which we explain in the sequel: (1) choices between t_i and t'_i or (2) sequentialised read access to p and p'.

For (1), notice that process i can produce r_i in two different ways. At least two occurrences of each r_i are thus present in the unfolding of any of the three nets. Hence there are at least 2^n ways of choosing t's preset, i.e. at least 2^n occurrences of t and p'' in any of the three unfoldings.

Roughly speaking, (2) refers to the same phenomena that were demonstrated in Fig. 2 and Fig. 3. While all t_i are concurrent in N_c, they are sequentialised in N_p: they all consume and produce the same p. This creates conflicts between them, and as a result all their exponentially many interleavings are explicitely present in $\mathcal{U}_{N_p}$. Importantly, any occurrence of t_i that consumes an occurrence of p at depth d, produces an occurrence of p at depth $d + 1$.

In N_r, even if all t_i are still concurrent to each other, their occurrences produce two conditions with occurrence depths 1 and 2, each labelled by their respective private copy of p. For $\mathcal{U}_{N_r}$, this again has the consequence of producing 2^n ways of choosing v_3's preset, and 2^n events labelled by v_3. More importantly, the private copies of t_i cannot be merged with those of t_j and they remain in $\mathcal{Q}_{N_r}$. As a result, all 2^n occurrences of v_3 are also present in the MP of N_r. This suggests that MPs of PR unfoldings may not yield, in general, much gain.

While the size of the contextual unfolding of N_c explodes due to (1), it is unaffected by (2). On the other hand, the MP of N_p effectively deals with (1), but

Table 1. Growth of the unfoldings and MPs of the n-GEN c-nets and their encodings.

Merged Processes			Unfoldings		
Ctx	Plain	PR	Ctx	Plain	PR
$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$

only partially with (2). We now see why. Notice that there are $\mathcal{O}(2^n)$ conditions labelled by p in $\mathcal{U}_{N_p}$, all with occurrence depths between 1 and $n+1$. In the MP, they are merged into the $n+1$ mp-conditions $p^1, \ldots, p^{n+1}$. Since all instances of q_i and r_i have occurrence-depth 1, all the exponentially many events labelled by t_i are merged into n mp-events, each consuming some p^j and producing p^{j+1}, for $1 \leq j \leq n$. This yields an MP of size $\mathcal{O}(n^2)$.

Finally, the CMP of N_c deals effectively with both (1) and (2); it is, in fact, isomorphic to N. Roughly speaking, this is because the unfolding of N_c already deals with (2), as we said, and the '*merging*' solves (1). Thus, the CMP is polynomially more compact than the MP of N_p and exponentially more than the MP of N_r, or the unfoldings of N_c, N_p, or N_r. See Table 1 for a summary.

While this example in itself is artificial, the underlying structures are quite simple and commonly occur in more complex c-nets, which explains some of the experimental results below.

5.2 Dijkstra's Mutual Exclusion Algorithm

In this section we analyse the performance of CMPs on a well-known concurrent algorithm for mutual exclusion due to Dijkstra [5]. What follows is a condensed technical explanation of the algorithm, see [5] for more details.

Dijkstra's algorithm allows n threads to ensure that no two of them are simultaneously in a critical section. Two Boolean arrays b and c of size n, and one integer variable k, satisfying $1 \leq k \leq n$, are employed. All the entries of both arrays are initialised to *true*, and k's initial value is irrelevant. All threads use the same algorithm, which runs in two phases. During the first, thread i sets $b[i] := \mathit{false}$, and repeatedly checks the value of $b[k]$, setting $k := i$ if $b[k]$ is true, until $k = i$ holds. At this point, thread i starts phase 2, where it sets $c[i] := \mathit{false}$, and enters the critical section if $c[j]$ holds for all $j \neq i$. If the check fails, it sets $c[i] := \mathit{true}$ and restarts in phase 1. After the critical section, $b[i]$ and $c[i]$ are set to *true*. Note that more than one thread could pass phase 1, and phase 2 is thus necessary.

We encoded Dijkstra's algorithm into a c-net as follows. The entries of arrays b, c are represented by two places, e.g. $b_i{=}t$ and $b_i{=}f$. Variable k is encoded by n places $k{=}0, \ldots, k{=}n{-}1$. Places $l_{0,i}, \ldots, l_{6,i}$ encode thread i's instruction pointer. Fig. 8 shows the fragment of 2-DIJKSTRA that encodes thread 0. Roughly, each transition encodes one instruction of the original algorithm [5], updating the instruction pointer and the variables affected by the instruction. Transitions

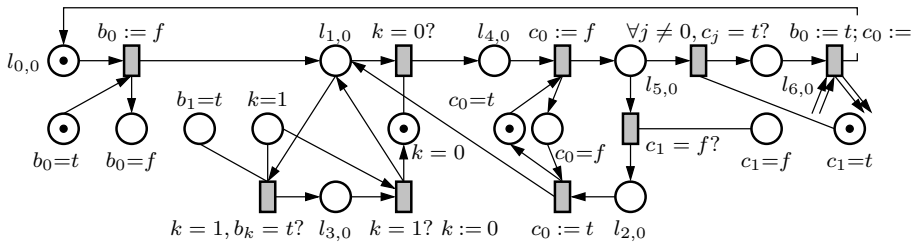

Fig. 8. The fragment of 2-DIJKSTRA that encodes thread 0. Note that arrows from transition $b_0 := t; c_0 := t$ are only partially depicted.

encoding conditional instructions, like $k = 0?$, or $\forall j \neq 0, c_j = t?$ employ read arcs to the places coding the variables involved in the predicate.

MPs of n-DIJKSTRA, and in particular CMPs, exhibit a very good growth with respect to n. Table 2 shows the figures, obtained under the same setting as in Section 5.3. While all unfoldings are exponential in n and $|T|$, all the MPs are of polynomial size. The sizes of the plain and PR unfoldings seem to increase by a factor of 5 for each process added. The contextual unfolding reduces this factor down to 3. The plain and PR MPs seems to fit a polynomial curve of degree close to 3. The CMP seems to grow linearly with n^2, i.e. linear with $|T|$, the number of transitions in the net. As it was the case for n-GEN, PR MPs seem to be less efficient than plain MPs on n-DIJKSTRA.

We note that this example exhibits some of the features explained in Section 5.1. For instance, process 0 can transition from $l_{5,0}$ to $l_{2,0}$ if there exists another process i with $c_i = f$. Thus, for $n \geq 3$ there would be a choice between multiple (i.e. $n-1$) transitions in parallel to implement the check, a structure also found in the n-GEN example. We note that such structures would also naturally ensue from other mutual exclusion algorithms that typically involve checking for the presence of some other event with a certain property.

Table 2. Unfolding and MP sizes of n-DIJKSTRA, its plain, and PR encodings. Last row obtained through regression analysis, see the text.

Net		Merged Processes			Unfoldings		
n	$\|T\|$	Ctx	Plain	PR	Ctx	Plain	PR
2	18	31	42	40	35	54	54
3	36	64	113	121	131	371	364
4	60	105	220	278	406	2080	1998
5	90	155	375	582	1139	10463	9822
6	126	214	589	1198	3000	49331	44993
	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(3^n)$	$\mathcal{O}(5^n)$	$\mathcal{O}(5^n)$

Table 3. Experimental results, see the text for more information.

Benchmark					Unfolding						Merged process						Gains		
Name	Net		Stats.		Plain		PR		Ctx		Plain		PR		Ctx				
	$\|P\|$	$\|T\|$	$\|C\|$	Repl.	$\|B\|$	$\|E\|^{(a)}$	$\|B\|$	$\|E\|$	$\|B\|$	$\|E\|^{(b)}$	$\|\widehat{B}\|$	$\|\widehat{E}\|^{(c)}$	$\|\widehat{B}\|$	$\|\widehat{E}\|$	$\|\widehat{B}\|$	$\|\widehat{E}\|^{(d)}$	a/d	b/d	a/c
Bds	53	59	24	15	4.50	3.79	2.60	2.14	424	252	1.11	1.14	1.36	1.07	70	44	21.73	5.73	19.12
Brujin	86	165	158	142	2.84	1.97	7.00	1.70	286	208	1.33	1.44	3.83	1.31	115	127	3.22	1.64	2.23
Byz	504	409	376	284	2.38	1.80	1.41	1.00	17019	7748	1.23	1.03	1.85	1.22	529	303	46.11	25.57	44.78
Eisenbahn	44	44	6	3	2.19	2.15	2.22	2.04	99	53	1.13	1.30	1.22	1.19	69	43	2.65	1.23	2.04
Ftp	176	529	39	33	1.06	1.04	1.02	1.00	73516	37540	1.02	1.05	1.19	1.00	254	455	85.74	82.51	81.61
Knuth	78	137	114	98	2.62	1.81	5.42	1.62	247	178	1.32	1.31	3.25	1.27	102	112	2.88	1.59	2.20
Mutual	49	41	12	4	1.41	1.23	1.51	1.23	187	121	1.12	1.26	1.23	1.27	92	73	2.04	1.66	1.62
Dme(2)	135	98	132	0	1.61	1.00	1.61	1.00	293	118	1.55	1.04	1.55	1.04	195	90	1.31	1.31	1.26
Dme(4)	269	196	264	0	1.73	1.00	1.73	1.00	1337	636	1.56	1.04	1.56	1.04	389	180	3.53	3.53	3.38
Dme(6)	403	294	396	0	1.79	1.00	1.79	1.00	3517	1794	1.56	1.04	1.56	1.04	583	270	6.64	6.64	6.36
Dme(8)	537	392	528	0	1.83	1.00	1.83	1.00	7217	3832	1.56	1.04	1.56	1.04	777	360	10.64	10.64	10.19
Dme(10)	671	490	660	0	1.86	1.00	1.86	1.00	12821	6990	1.56	1.04	1.56	1.04	971	450	15.53	15.53	14.87
Elev(1)	63	99	18	11	1.02	1.00	1.34	1.00	152	85	1.02	1.00	1.38	1.00	56	46	1.85	1.85	1.85
Elev(2)	146	299	59	47	1.03	1.00	1.85	1.00	858	477	1.01	1.00	2.02	1.00	125	135	3.53	3.53	3.53
Elev(3)	327	783	160	141	1.03	1.00	2.66	1.00	4050	2241	1.00	1.00	2.90	1.00	263	346	6.48	6.48	6.48
Elev(4)	736	1939	405	375	1.04	1.00	4.08	1.00	17360	9567	1.00	1.00	3.98	1.00	556	841	11.38	11.38	11.38
Furn(1)	27	37	9	5	1.12	1.04	1.18	1.00	130	72	1.09	1.03	1.23	1.00	43	33	2.27	2.18	2.21
Furn(2)	40	65	11	6	1.14	1.06	1.14	1.00	582	324	1.07	1.05	1.19	1.02	75	94	3.66	3.45	3.47
Furn(3)	53	99	13	7	1.14	1.08	1.12	1.00	2265	1250	1.07	1.02	1.17	1.01	106	221	6.09	5.66	5.95
Key(2)	94	92	32	31	2.74	2.16	1.72	1.27	302	191	1.22	2.50	1.59	1.17	112	105	3.92	1.82	1.57
Key(3)	129	133	48	47	5.81	4.60	2.81	2.19	1276	806	1.21	4.13	1.64	2.34	151	186	19.93	4.33	4.83
Key(4)	164	174	64	63	11.37	9.08	5.56	4.43	5806	3637	1.21	5.26	1.67	9.92	190	290	113.82	12.54	21.63
Mmgt(1)	50	58	1	0	1.00	1.00	1.00	1.00	79	38	1.00	1.00	1.00	1.00	57	38	1.00	1.00	1.00
Mmgt(2)	86	114	2	0	1.00	1.00	1.00	1.00	502	250	1.00	1.00	1.00	1.00	99	155	1.61	1.61	1.61
Mmgt(3)	122	172	3	0	1.00	1.00	1.00	1.00	2849	1424	1.00	1.00	1.00	1.00	141	355	4.01	4.01	4.01
Mmgt(4)	158	232	4	0	1.00	1.00	1.00	1.00	14900	7450	1.00	1.00	1.00	1.00	183	638	11.68	11.68	11.68
RW(1,1)	84	208	123	75	1.23	1.00	1.78	1.00	142	94	1.26	1.00	2.03	1.00	77	65	1.45	1.45	1.45
RW(2,1)	72	88	27	16	1.19	1.01	1.30	1.00	845	554	1.26	1.01	1.45	1.00	113	165	3.39	3.36	3.37
RW(3,1)	106	270	129	81	1.19	1.04	1.60	1.00	5100	3376	1.24	1.02	1.93	1.00	160	368	9.54	9.17	9.39
RW(1,2)	209	1482	1132	717	1.21	1.00	3.36	1.00	2836	1838	1.35	1.01	4.76	0.99	159	371	4.95	4.95	4.90
Sentest(25)	104	55	8	5	1.32	1.29	1.35	1.29	188	104	1.06	1.11	1.10	1.11	107	55	2.44	1.89	2.20
Sentest(50)	179	80	8	5	1.23	1.23	1.25	1.23	263	129	1.03	1.08	1.06	1.08	182	80	1.99	1.61	1.85
Sentest(75)	254	105	8	5	1.18	1.19	1.19	1.19	338	154	1.02	1.06	1.04	1.06	257	105	1.75	1.47	1.66
Sentest(100)	329	130	8	5	1.15	1.17	1.16	1.17	413	179	1.02	1.05	1.03	1.05	332	130	1.61	1.38	1.54

5.3 Assorted Benchmarks

In this section we present experimental results for a number of benchmark examples circulating in the PN community (collected mostly by Corbett [4]). The following consistent setup was used to produce them:

- The total adequate order proposed in [11] was used.
- All configurations were allowed as cutoff correspondents.
- The cutoff (mp-)events and post-cutoff (mp-)conditions were not counted.

The plain and PR unfolding prefixes were constructed using Punf [12], and the contextual unfolding prefixes were computed by compressing the PR ones with PRCompress[6]. The plain and PR MPs have been merged from the corresponding unfolding prefixes with Mci2mp, and the CMPs were merged from the corresponding contextual unfolding prefixes using Cmerge. Note that the direct construction of contextual unfoldings and MPs would yield the same results [11, 1].

Recall the following theoretical guarantees:

- The contextual unfolding prefix is never larger than the PR prefix.
- The plain/PR/contextual MP is never larger than the corresponding unfolding prefix.

Table 3 compares the sizes of plain, PR and contextual unfolding prefixes and MPs. The 4^{th} and 5^{th} columns from the left are, respectively, the number of read arcs in the net and place replicas in its PR encoding.[7] The number of conditions and events for the plain and PR unfoldings is normalised wrt. that of the contextual unfolding. Similarly, mp-conditions and mp-events of the plain and PR MPs are normalised wrt. those of the CMPs. The last three columns show the compression gains of CMPs wrt. plain and contextual unfolding prefixes, and the gain of plain MPs wrt. plain unfolding prefixes.

One can see that CMPs are the most compact of all the considered representations.[8] Furthermore, on some benchmarks, notably Key(4), it has significant advantages over both plain and PR MPs. Interestingly, in this case the PR MP is significantly larger than even the plain MP, which seems to be due to place replication making the subsequent merging much less efficient. As CMPs do not suffer from this problem, they come as a clear winner in such cases.

6 Conclusions and Future Work

We have developed a new condensed representation of the state space of a contextual Petri net, called contextual merged processes. This representation combines the advantages of merged processes and contextual unfoldings, and copes

[6] All tools available from the URL indicated in Footnote 5.

[7] More precisely, $\sum_{p \in P, |\underline{p}|>1}(|\underline{p}| - 1)$.

[8] Though the PR MP of RW(1,2) has four mp-events fewer, it has many more mp-conditions.

with several important sources of state space explosion: concurrency, sequences of choices, and concurrent read accesses to shared resources. The experimental results demonstrate that this representation is significantly more compact than either merged processes or contextual unfoldings.

We also proved a number of results which lay the foundation for model checking of reachability-like properties of safe c-nets based on CMPs. In particular, given a CMP, they allow one to reduce (in polynomial time) such a model checking problem to SAT. Furthermore, since the algorithm for direct construction of merged processes of safe Petri nets proposed in [11] is based on model checking, it can be transferred to the contextual case, which would complete the verification flow based on CMPs.

We currently work on implementing the proposed model checking algorithm and on porting the algorithm for direct construction of MPs proposed in [11] to the contextual case. (While the high-level structure of the latter algorithm remains the same, moving from Petri nets to c-nets entails several low-level changes in the nets representation, which pervade the whole code; thus, this porting requires significant implementation effort.)

Another possible direction of future work is to generalise our approach. Normal Petri net unfoldings work very well when systems are entirely concurrent and independent of one another, but many sources of state-space explosion appear when they interact. The approaches that we have combined in this work tackle two such sources; they compress the unfolding and have further commonalities. While Petri net unfoldings are *structurally* acyclic, c-net unfoldings and merged processes have structural cycles but could be said to be *semantically* acyclic: every marking can still be reached by a repetition-free execution and hence one retains the NP-completeness of reachability problem (which is PSPACE-complete for safe Petri nets). This poses the question whether our solutions are a part of a more general phenomenon. The following example suggests that this might be the case. Consider Fig. 9 (a). The token on place p acts as a lock ensuring mutual exclusion between two critical sections represented by places b_1 and b_2.

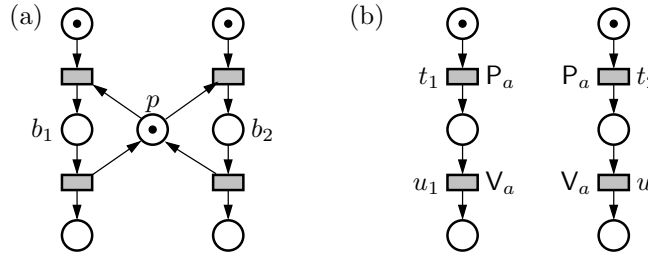

Fig. 9. Two processes competing for lock p: (a) a Petri net (b) a net where lock operations are annotated on transitions.

The two processes are independent of one another, except for the temporal restriction that they cannot possess the lock p at the same time. This imposes a truly semantic sequentialisation constraint (unlike the sequentialisation in Fig. 3 (b), which is merely due to an inadequate semantics-changing encoding). The traditional unfolding techniques cannot take advantage of the fact that the processes are otherwise independent. Indeed, when the example from Fig. 9 is scaled to n processes, a complete unfolding prefix is of size $\mathcal{O}(2^n)$ and a complete MP is still of size $\mathcal{O}(n^2)$ when produced by the tool PUNF.

It is conceivable that this case could be handled by treating locks explicitly and annotating transitions with locking (P) and unlocking (V) actions, like in Fig. 9 (b). When multiple locks are involved, their use may introduce circular precedence constraints that can be captured with, e.g. the Lock Causality Graphs of [10]. A suitably defined unfolding for such a case would then unfold both processes independently, only demanding that configurations do not include circular lock constraints. One easily observes that in such a setting, like in ours, an event may have multiple histories that would need to be taken into account to determine cutoffs. In Fig. 9 (b), for instance, t_2 may occur either individually or in a context in which t_1, u_1 *must* have occurred before it. It is therefore quite conceivable that locks could be seamlessly integrated with CMPs as they once again exhibit similar characteristics. To conclude, an interesting perspective for the future research would be to develop a generic framework that handles such effects.

References

1. Baldan, P., Bruni, A., Corradini, A., König, B., Rodríguez, C., Schwoon, S.: Efficient unfolding of contextual Petri nets. TCS 449, 2–22 (2012)
2. Baldan, P., Corradini, A., König, B., Schwoon, S.: McMillan's complete prefix for contextual nets. ToPNoC 1, 199–220 (2008), LNCS 5100
3. Baldan, P., Corradini, A., Montanari, U.: Contextual Petri nets, asymmetric event structures, and processes. Inf. Comput. 171(1), 1–49 (2001)
4. Corbett, J.C.: Evaluating deadlock detection methods for concurrent software. IEEE Transactions on Software Engineering 22, 161–180 (1996)
5. Dijkstra, E.W.: Solution of a problem in concurrent programming control. Commun. ACM 8(9), 569ff (Sep 1965)
6. Esparza, J., Römer, S., Vogler, W.: An improvement of McMillan's unfolding algorithm. Formal Methods in System Design 20, 285–310 (2002)
7. Heljanko, K.: Deadlock and reachability checking with finite complete prefixes. Licentiate's thesis, Helsinki University of Technology (1999)
8. Heljanko, K.: Minimizing finite complete prefixes. In: Proc. CS&P. pp. 83–95 (1999)
9. Janicki, R., Koutny, M.: Invariant semantics of nets with inhibitor arcs. In: Proc. CONCUR. LNCS, vol. 527, pp. 317–331 (1991)
10. Kahlon, V.: Boundedness vs. unboundedness of lock chains: Characterizing decidability of CFL-reachability for threads communicating via locks. In: Proc. LICS. pp. 27–36 (2009)
11. Khomenko, V., Mokhov, A.: An algorithm for direct construction of complete merged processes. In: Proc. Petri Nets. pp. 89–108. LNCS 6709 (2011)

12. Khomenko, V.: PUNF, homepages.cs.ncl.ac.uk/victor.khomenko/tools/punf/
13. Khomenko, V.: Model Checking Based on Prefixes of Petri Net Unfoldings. Ph.D. thesis, School of Computing Science, Newcastle University (2003)
14. Khomenko, V., Kondratyev, A., Koutny, M., Vogler, W.: Merged processes – a new condensed representation of Petri net behaviour. Act. Inf. 43(5), 307–330 (2006)
15. McMillan, K.L.: Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In: Proc. CAV. pp. 164–177. LNCS 663 (1992)
16. Montanari, U., Rossi, F.: Contextual occurrence nets and concurrent constraint programming. In: Dagstuhl Seminar 9301. LNCS, vol. 776, pp. 280–295 (1994)
17. Ranjan, D.P., Tang, D., Malik, S.: A comparative study of 2QBF algorithms. In: Proc. SAT (2004)
18. Ristori, G.: Modelling Systems with Shared Resources via Petri Nets. Ph.D. thesis, Department of Computer Science, University of Pisa (1994)
19. Rodríguez, C.: CUNF, http://www.lsv.ens-cachan.fr/~rodriguez/tools/cunf/
20. Rodríguez, C., Schwoon, S.: Verification of Petri Nets with Read Arcs. In: Proc. CONCUR. LNCS, vol. 7454, pp. 471–485 (Sep 2012)
21. Rodríguez, C., Schwoon, S., Baldan, P.: Efficient contextual unfolding. In: Proc. CONCUR. LNCS, vol. 6901, pp. 342–357 (Sep 2011)
22. Rodríguez, C., Schwoon, S., Khomenko, V.: Contextual merged processes. Tech. Rep. LSV-13-06, LSV, ENS de Cachan, France (2013)
23. Valmari, A.: The state explosion problem. In: Lectures on Petri Nets I: Basic Models, LNCS, vol. 1491, pp. 429–528. Springer, Berlin Heidelberg (1998)
24. Vogler, W., Semenov, A.L., Yakovlev, A.: Unfolding and finite prefix for nets with read arcs. In: Proc. CONCUR. LNCS, vol. 1466, pp. 501–516 (1998)