Unfolding and Finite Prefix for Nets with Read Arcs

Walter Vogler*
Institut für Informatik, Universität Augsburg
D-86135 Augsburg, Germany
email: vogler@informatik.uni-augsburg.de

Alex Semenov and Alex Yakovlev †
Department of Computing Science
University of Newcastle upon Tyne, NE1 7RU, U.K.
email: alex.yakovlev@ncl.ac.uk

Technical Report Series No. 634
Computing Science
University of Newcastle upon Tyne

Abstract

Petri nets with read arcs are investigated with respect to their unfolding, where read arcs model reading without consuming, which is often more adequate than the destructive-read-and-rewrite modelled with loops in ordinary nets. The paper redefines the concepts of a branching process and unfolding for nets with read arcs and proves that the set of reachable markings of a net is completely represented by its unfolding. The specific feature of branching processes of nets with read arcs is that the notion of a co-set is no longer based only on the binary concurrency relation between the elements of the unfolding, contrary to ordinary nets. It is shown that the existing conditions for finite prefix construction (McMillan's one and its improvement by Esparza et al.) can only be applied for a subclass of nets with read arcs, the so-called read-persistent nets. Though being restrictive, this subclass is sufficiently practical due to its conformance to the notion of hazard-freedom in logic circuits. The latter appear to be one of the most promising applications for nets with read arcs.
Keywords: asynchronous circuits, branching processes, concurrency semantics, hazards, Petri nets with read arcs, read-persistence, unfolding

1 Introduction

Use of partial order semantics in the analysis of Petri net models of concurrent systems often appears to be the only practical alternative due to combinatorial explosion of the model state space. The unfolding of a Petri net is a way to represent partial order or

*Work on this paper was partially supported by the DFG (Project 'Halbordnungstesten').
†Partially supported by EPSRC (projects HADES and TIMBRE, GR/K70175/L28098)

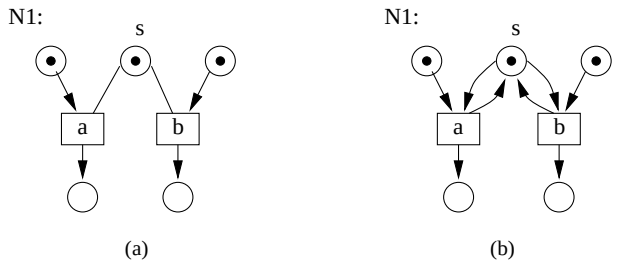

Figure 1 Example of a Petri net with read arcs (a) and its ordinary Petri net "equivalent" (b).

process semantics in terms of the Petri net model. The unfolding net contains information about all reachable markings and firable transitions of its generator net. Furthermore, for a bounded Petri net one should only consider an initial fragment of the unfolding to capture this information. McMillan [McM93] proposed an original algorithm of constructing such a finite prefix representing the behaviour of an ordinary place-transition Petri net.

Recently, Petri nets with read arcs have found considerable interest [CH93, JK95, MR95, BG95, BP96]; read arcs – as the lines from s in Figure 1(a) – describe reading without consuming, e.g. reading in a database; consequently, a and b in N_1 can occur concurrently. In ordinary nets (cf. Figure 1(b)), loops (arcs from a to s and from s to a and similarly for b) would be used instead, which describe a destructive-read-and-rewrite and do not allow concurrency; this is certainly not always adequate.

A number of application areas call for the use of Petri nets with read arcs. One example is the modelling of concurrent programs with synchronisation mechanisms, where some places represent binary control variables which can be "tested" by some actions and "set", incremented or decremented, by other actions (cf. [EB96]).

Another example is the modelling and analysis of asynchronous logic circuits, in particular checking whether their behaviour may contain hazards (potential deviations of the circuit behaviour from the specification due to the fact that an output signal transition can be non-deterministically disabled by an input signal change). One of the common techniques [YKSK96] to model logic circuits is based on associating each gate in a circuit with a special Petri net fragment as shown in Figure 2. Each input or output signal is modelled by a pair of complementary places, one for state 0 and the other for state 1. The up and down going transitions of the output are modelled by Petri net transitions labelled with $d+$ and $d-$ (sometimes several copies are needed). These transitions are connected to the places corresponding to the state of the output by ordinary consuming and producing arcs and to the places representing the state of inputs by read arcs. The latter type manifests the fact that the gate's action cannot change the state of the gate's inputs. Each input is controlled either by transitions of some other gate, whose output is connected to this input, or by transitions in the environment. The subsequent construction of the circuit Petri net (with read arcs) model from such fragments is quite obvious. For example, let the output of gate g_1 be connected to an input of gate g_2. Then, the places corresponding to the output of gate g_1 are merged with the places corresponding to the given input of gate g_2. Those places are then connected by read arcs to the transitions of the g_2. If there are several gates whose inputs are connected to the same output, the corresponding net will have places connected to multiple reader transitions. This allows modelling concurrent (non-destructive) reading of the same condition by multiple transitions.

It should be clear that the idea of modelling logic circuits with Petri nets with read

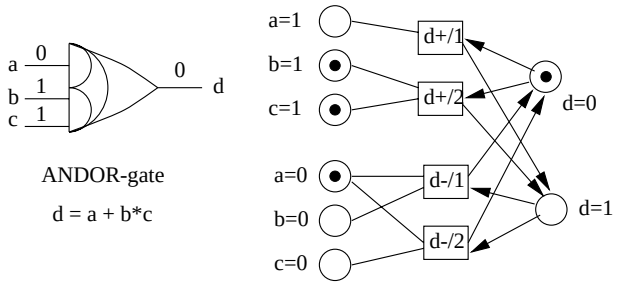

Figure 2 Logic gate (a) and corresponding Petri net fragment with read arcs (b).

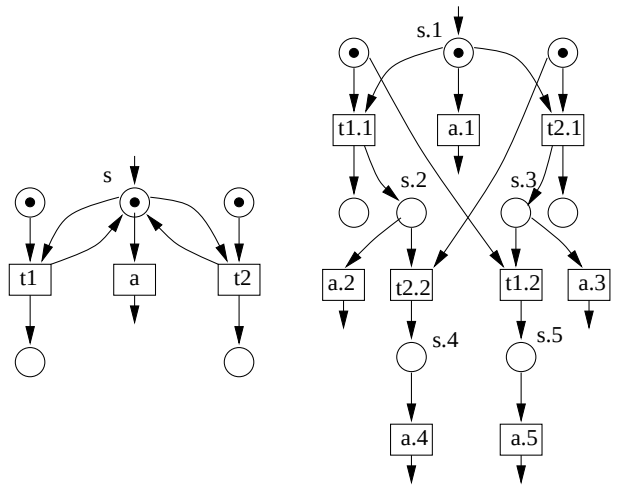

Figure 3 Fragment of Petri net with loops and its unfolding fragment.

arcs can be extended to modelling any discrete event structures built of components with unidirectional interconnections; where each component can be represented by a Petri net fragment of the above-mentioned type.

A Petri net with read arcs produces exactly the same behaviour in terms of its interleaving semantics (reachable markings and firing sequences) as its ordinary Petri net "equivalent" wherein loops replace read arcs. It can therefore be analysed using the already existing unfolding mechanisms, e.g. McMillan's one [McM93]. However, examples show that the size of the unfolding can grow very fast for nets with a high out-degree of places with loops. The major reason for that is that such a place is regarded in McMillan's unfolding as a conflict one and thus it produces a combinatorial set of alternative serialized executions. It is easy to see that if a place s has n loops connected to transitions $t_1, ..., t_n$ as illustrated for $n = 2$ in Figure 3, the unfolding will have $n!$ paths, producing in total $K_n = \sum_{r=1}^{n-1} n!/(n-r)!$ loop transition instances and $K_n + 1$ instances of place s. If s has n read arcs instead of loops, the net (e.g. a modification of Figure 3) will turn out to be its own unfolding, i.e. there are n read transition instances and one instance of s.

A special technique based on loops and replication of the read place s can be applied to the net with read arcs before it is unfolded (cf. Figure 4), such that each potential 'reader' of s gets its own copy. This technique helps avoid combinatorial explosion of transitions $t_1, ..., t_n$ and place s. However, if the token in the read place s can be consumed by another (firable) transition a, the above-mentioned problem "shifts" to this transition; it will produce 2^n instances in the unfolding, whereas there is only one if read arcs are used.

These simple examples suggest that significant savings in the size of the unfolding prefix

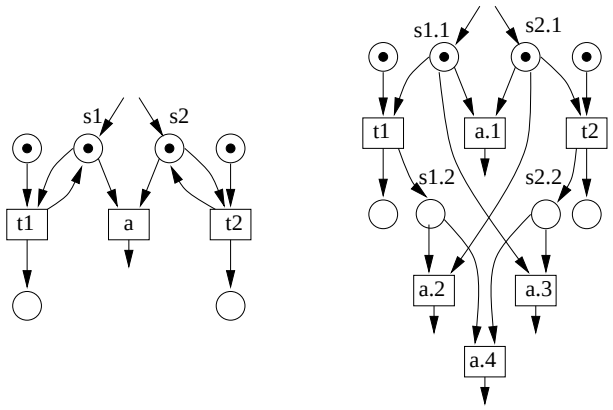

Figure 4 Illustration of loop and place replication technique and its unfolding.

could be achieved if the original Petri net with read arcs was unfolded directly, preserving the notion of read arc in its process semantics.

[MR95, JK95, BP96, Vog97] define processes of nets with read arcs. We extend here the theory of partial order semantics for nets with read arcs by defining the new concepts of a conditional precedence relation, a contextual cycle and a co-set (Section 2). For example, a co-set, i.e. a set of elements that are independent or concurrent, for a net with read arcs cannot be generalised from the pair-wise concurrency relation, the property customary to ordinary nets.

Based on the new idea of a co-set, we redefine the notions of occurrence net, branching process and unfolding (Section 3). We then (Section 4) prove the fact that the unfolding contains the full information about reachability of a net with read arcs, for which we need new definitions of a configuration and a process induced by a configuration, and properties relating cuts in the unfolding and reachable markings of the original net.

As in the ordinary setting, our unfolding is an overlay of the processes defined in [MR95, JK95, Vog97]; but here, an event may have different 'local configurations' in the processes it belongs to. In order to be able to apply the existing unfolding truncation (or cutoff) techniques from [McM93, ERV96], at the stage of proving the completeness of a finite unfolding prefix (Section 5), we restrict ourselves to a subclass of nets with read arcs, called read-persistent nets. We prove that the usual prefix completeness is satisfied for such nets. We also demonstrate their practical usefulness and savings achievable, in terms of the size of the unfolding, when compared to "equivalent" modelling with loops in ordinary nets.

The main contribution of the paper is therefore twofold: (1) theoretical framework for the branching processes and unfolding of nets with read arcs; and (2) the algorithm for constructing a finite unfolding prefix for a practically useful subclass of nets with read arcs (we also demonstrate the problem that arises for nets with read arcs in general).

2 Petri nets, read arcs and occurrence nets

In this section, we introduce safe Petri nets (place/transition-nets) with read arcs. Read arcs are also called positive contexts [MR95], test arcs [CH93] or activator arcs [JK95]. Furthermore, we introduce a suitable notion of occurrence net which will serve as the basis to define the unfolding of a net.

We start with some relational notions: for a relation $R \subseteq X \times X$, we often write xRy

in lieu of $(x,y) \in R$. Composition of relations on X is defined by $R \circ S = \{(x,z) \mid \exists y \in X : (x,y) \in R \wedge (y,z) \in S\}$. We write R^+ for the transitive closure of R, and R^{-1} for its inverse. Assume $\sqsubset$ is a *partial order* on X, i.e. it is irreflexive and transitive. A *linearization* of $\sqsubset$ is a sequence containing each element of X once such that x occurs before y whenever $x \sqsubset y$.

A *net graph* (with read arcs) (S,T,W,R) consists of disjoint sets S of *places* and T of *transitions*, the (ordinary) *arcs* $W \subseteq S \times T \cup T \times S$ (which all have weight 1) and the set of *read arcs* $R \subseteq S \times T$, where we always assume $(R \cup R^{-1}) \cap W = \emptyset$. As usual, we draw transitions as boxes, places as circles and arcs as arrows; read arcs are drawn as lines without arrow heads.

For each $x \in S \cup T$, the *preset* of x is ${}^\bullet x = \{y \mid (y,x) \in W\}$, the *read set* of x is $\hat{x} = \{y \mid (y,x) \in R \cup R^{-1}\}$, and the *postset* of x is $x^\bullet = \{y \mid (x,y) \in W\}$; furthermore, the *preconditions* of $t \in T$ are $pre(t) = {}^\bullet t \cup \hat{t}$, those of $s \in S$ are $pre(s) = {}^\bullet s$.[1] These notions are extended pointwise to sets, e.g. ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$. If $x \in {}^\bullet y \cap y^\bullet$, then x and y form a *loop*. We only consider net graphs (and nets) which are *T-restricted*, i.e. satisfy ${}^\bullet t \neq \emptyset \neq t^\bullet$ for all $t \in T$.

A *marking* is a function $S \to \mathbb{N}_0$. A *Petri net with read arcs* $N = (S,T,W,R,M_N)$ (or just a *net* for short) consists of a finite net graph and an *initial marking* $M_N : S \to \{0,1\}$. When we introduce a net N or N_1 etc., then we assume that implicitly this introduces its components S, T, W, ... or S_1, T_1, ..., etc. and similarly for other tuples later on. A net is called *ordinary*, if $R = \emptyset$. We sometimes regard sets as characteristic functions, which map the elements of the sets to 1 and are 0 everywhere else; hence, we can e.g. add a marking and a postset of a transition or compare them componentwise. Vice versa, a function with images in $\{0,1\}$ is sometimes regarded as a set such that we can e.g. apply union to it.

The net graph (S',T',W',R') is a *subnet* of the net graph (S,T,W,R) if $S' \subseteq S$, $T' \subseteq T$ etc.; it is *induced* by $S' \cup T'$, if $R' = R \cap S' \times T'$ and similarly for W'.

We now define the basic firing rule, which extends the firing rule for ordinary nets by regarding the read arcs as loops.

- A transition t is *enabled* under a marking M, denoted by $M[t\rangle$, if $pre(t) \leq M$.

  If $M[t\rangle$ and $M' = M + t^\bullet - {}^\bullet t$, then we denote this by $M[t\rangle M'$ and say that t can *occur* or *fire* under M yielding the marking M'. Thus, when t fires, it checks its pre- and read-set, removes a token from each place in its preset and puts a token onto each place in its postset.

- This definition of enabling and occurrence can be extended to sequences as usual: a sequence w of transitions is *enabled* under a marking M, denoted by $M[w\rangle$, and yields the follower marking M' when *occurring*, denoted by $M[w\rangle M'$, if $w = \lambda$ and $M = M'$ or $w = w't$, $M[w'\rangle M''$ and $M''[t\rangle M'$ for some marking M''. If w is enabled under the initial marking, then it is called a *firing sequence*.

  A marking M is called *reachable* if $\exists w \in T^* : M_N[w\rangle M$. The net is *safe* if $M(s) \leq 1$ for all places s and reachable markings M.

General assumption All nets considered in this paper are *safe* and, as already stated, *finite* and *T-restricted* (where we sometimes omit the postsets of transitions in figures).

[1] We will use $pre(s)$ for a place s when this set is empty or consists of one transition whose firing is indeed a precondition for s being marked.

Now we will define occurrence nets for Petri nets with read arcs in the sense of [NPW81], i.e. for the description of concurrency *and* choice in the runs of a net.

An occurrence net is a net graph $O = (B, E, F, A)$ satisfying some requirements listed below; we call $b \in B$ a *condition* and $e \in E$ an *event*. (The letter A is used since read arcs are called activator arcs in [JK95].) On $B \cup E$, we define *causality* $<$ as $(F \cup A)^+$ and write $x \leq y$ for $x < y \vee x = y$. We could also call $<$ unconditional causality, since intuitively $x < y$ will mean that x necessarily occurs (i.e. fires or is created) before y.

Let $X \subseteq B \cup E$. The *causal closure* of X is $\downarrow X = \{y \mid \exists x \in X : \ y \leq x\}$. We call X *causally closed*, if $x < y \in X$ implies $x \in X$.

The *precedence relation* $\sqsubset_X$ (or just $\sqsubset$ if X is clear from the context) on X is $(F_X \cup A_X \cup A_X^{-1} \circ F_X)^+$, where A_X and F_X are the restrictions of A and F to X. We could call $\sqsubset$ conditional causality, since its intuitive meaning is: *if* all elements of X occur, they occur in an order obeying $\sqsubset$.

These intuitive explanations could be made precise, if we mark the minimal conditions and fire transitions also in O, compare e.g. [Eng91, Vog91]; see the comments on Figure 5 below.

A *contextual cycle* in X is a cycle with edges in $F_X \cup A_X \cup A_X^{-1} \circ F_X$; thus, X has no contextual cycle if and only if $\sqsubset_X$ is a partial order. $X \subseteq B$ is a *co-set*, if

- $\downarrow X$ is *conflict-free*, i.e. all events e_1, e_2 in $\downarrow X$ satisfy ${}^\bullet e_1 \cap {}^\bullet e_2 \neq \emptyset \Rightarrow e_1 = e_2$.
- $\downarrow X$ has no contextual cycle, i.e. $\sqsubset_X$ is a partial order.
- X *may coexist*, i.e. all conditions $b \in X$ satisfy $b^\bullet \cap \downarrow X = \emptyset$.

A *cut* is a maximal co-set.

The causal closure is formed by closing under *pre*, so in some ways the *preconditions* play the rôle of the presets in ordinary nets. On the other hand, the first requirement above for a co-set is the usual one, i.e. it uses the *preset*. Also, the third requirement is (a reformulation of) the usual requirement, saying that b is not in the *preset* of an event in $\downarrow X$; note that causally related conditions can coexist in an occurrence net (cf. the net of Figure 5, which after removal of $e4$, $b7$ and the tokens is an occurrence net; here, e.g., $b1$ coexists with $b6$ reflecting part of the marking reached by firing $e3$). The second requirement is indigenous to the setting with read arcs and its correct formulation is one of the main contributions of this paper; due to this requirement, co-sets cannot be defined based on a binary co-relation, which is possible in the ordinary setting. The example shown in Figure 5 illustrates this effect. Namely, here are pairwise co-sets: $\{b4, b5\}, \{b4, b6\}$ and $\{b5, b6\}$, which however do not produce a co-set consisting of all three conditions; events $e1, e2$ and $e3$ form a contextual cycle. Thus, event $e4$ can never be enabled.

Note that in order to mark $b4$ or $b6$, we *have* to fire $e1$ or $e3$ (unconditional causality). We can fire $e1$ immediately, but *if* we want to fire $e1$ *and* $e3$, we have to fire $e3$ first (conditional causality).

Now O is an *occurrence net* if

- $\forall b \in B : \ |{}^\bullet b| \leq 1$;
- O is *finitary*, i.e. $\forall x \in B \cup E : \downarrow \{x\}$ is finite;
- $\forall e \in E$: $pre(e)$ is a co-set.

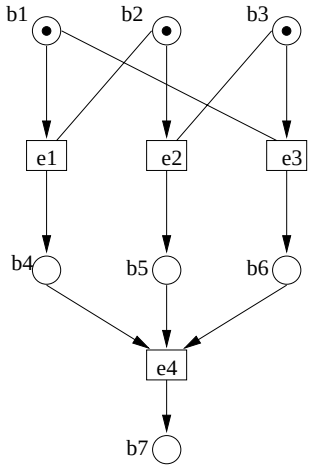

Figure 5 Example illustrating the notions of co-set and contextual cycle.

Furthermore, for an occurrence net O, we define $Min(O) = \{b \in B \mid {}^\bullet b = \emptyset\}$. Since all our net graphs are T-restricted, this is the set of minimal elements of $(B \cup E, <)$.

Our first two requirements for an occurrence net are usual; the third one corresponds to the usual requirement that no event be in 'self-conflict'. Usually, an occurrence net is required to be acyclic. Similarly, we have:

Proposition 2.1 *For an occurrence net O, $<$ is a partial order on $B \cup E$, i.e. $F \cup A$ is an acyclic graph.*

Proof: If $F \cup A$ had a cycle, this would contain an event e, and then it would be a contextual cycle in $\downarrow pre(e)$, a contradiction. □

By the similarity to the ordinary setting, it is easy to believe statements that are in fact wrong in our setting with read arcs; see Lemma 5.2 below as an example for a statement that sounds plausible and holds for ordinary nets, but fails for general nets with read arcs. We therefore try to be careful in formulating also small steps; e.g. we have:

Proposition 2.2 *Let O be an occurrence net. Then*

i) A subset of a co-set is a co-set.

ii) A subnet O' of O is an occurrence net; if $X \subseteq B' \cup E'$ is a co-set in O, it is also one in O'.

Proof: Part i) is obvious since for $X \subseteq Y$ also $\downarrow X \subseteq \downarrow Y$. Also, $\downarrow X$ formed in a subnet O' is contained in $\downarrow X$ formed in O; since pre- and postsets can only be smaller in O' and O' cannot have additional contextual cycles, the second statement of Part ii) follows, and with it also the first one. □

3 Branching processes and unfolding

We will now develop the theory of branching processes and unfoldings of nets with read arcs, following more or less the treatment of ordinary nets in [Eng91], and provide results that can be used to show the correctness of the finite prefix algorithm as in [ERV96].

For net graphs N_1 and N_2, a *homomorphism* from N_1 to N_2 is a mapping $h : S_1 \cup T_1 \to S_2 \cup T_2$ with

- $h(S_1) \subseteq S_2$, $h(T_1) \subseteq T_2$;
- for all $t \in T$, h is injective on ${}^\bullet t$, $\hat{t}$ and $t^\bullet$, and maps these to ${}^\bullet h(t)$, $\hat{h}(t)$ and $h(t)^\bullet$.

Definition 3.1 A *branching process* or *b-process* (O, p) of a net N consists of an occurrence net O and a homomorphism p from O to (the net graph of) N such that

i) p is injective on $Min(O)$ with $p(Min(O)) = M_N$;

ii) $\forall e_1, e_2 \in E : pre(e_1) = pre(e_2) \wedge p(e_1) = p(e_2) \Rightarrow e_1 = e_2$.

□

Remark: In this context, $pre(e_1) = pre(e_2)$ is equivalent to ${}^\bullet e_1 = {}^\bullet e_2 \wedge \hat{e}_1 = \hat{e}_2$, since the labelling of e_i and $b \in pre(e_i)$ determines whether $b \in {}^\bullet e_i$ or $b \in \hat{e}_i$. □

Two b-processes (O_1, p_1) and (O_2, p_2) are *isomorphic* if there is a bijective homomorphism h from O_1 to O_2 such that $\forall x \in B_1 \cup E_1 : p_2(h(x)) = p_1(x)$.

A b-process (O, p) is *canonical*, if $x = (p(x), pre(x))$ for all $x \in B \cup E$; i.e. the identities of the elements of O determine p and the graph with edges $F \cup A$.

Note that Definition 3.1 ii) also holds for conditions b_1 and b_2: if $pre(b_1) = pre(b_2)$, then either $b_1, b_2 \in Min(O)$ or b_1 and b_2 are in the postset of the same event; now by 3.1 i) or by definition of a homomorphism, $p(b_1) = p(b_2)$ implies $b_1 = b_2$. With this extension of 3.1 ii) and since O is finitary and $F \cup A$ acyclic by 2.1, it is easy to show the following theorem; compare [Eng91], in particular the last paragraph of Section 4.

Theorem 3.2 *Each b-process is isomorphic to a unique canonical b-process.*

Analogously to [Eng91, Lemma 20], we have:

Lemma 3.3 *Let (O, p) be a canonical b-process.*

i) $\forall (s, P) \in B : \ pre(s, P) = P \wedge |P| \leq 1$

ii) $\forall (t, P) \in E : \ pre(t, P) = P \wedge (t, P)^\bullet = \{(s, \{(t, P)\}) \mid s \in t^\bullet\}$

iii) $Min(O) = \{(s, \emptyset) \mid s \in M_N\}$

iv) $\forall (x, P) \in B \cup E : \ p(x, P) = x$

Definition 3.4 We call (O_1, p_1), where O_1 is a net graph and $p_1 : B_1 \cup E_1 \to S \cup T$ a mapping, a *prefix* of a b-process (O_2, p_2) of a net N if O_1 is a subnet of O_2 such that

i) $Min(O_2) \subseteq B_1$;

ii) for all $b \in B_1$, the event in ${}^\bullet b$ in O_2 (if it exists) belongs to E_1;

iii) for all $e \in E_1$, the conditions in ${}^\bullet e \cup \hat{e} \cup e^\bullet$ formed in O_2 belong to B_1;

iv) p_1 is the appropriate restriction of p_2.

□

Lemma 3.5 *If (O_1, p_1) is a prefix of (O_2, p_2), then $Min(O_1) = Min(O_2)$.*

Proof: If $b \in Min(O_1)$, then clearly $b \in B_2$ and ${}^\bullet b = \emptyset$ in O_2 by ii) above. Vice versa, if $b \in Min(O_2)$, then $b \in B_1$ by i) above and ${}^\bullet b = \emptyset$ in O_1, since O_1 is a subnet of O_2. □

We now show a slightly stronger variation of [Eng91, Lemma 21].

Lemma 3.6 *Let (O_2, p_2) be a canonical b-process of a net N. (O_1, p_1) is a prefix of (O_2, p_2) if and only if (O_1, p_1) is a canonical b-process of N, $B_1 \subseteq B_2$ and $E_1 \subseteq E_2$.*

Proof: (if) By Lemma 3.3 i) and ii), O_1 is a subnet of O_2 – note that by ii) an event $e \in E_1$ has the same set $pre(e)$ in both b-processes and hence the same sets ${}^\bullet e$ and $\hat{e}$ since these can be separated by the labelling. By Lemma 3.3 iii) $Min(O_2) = Min(O_1) \subseteq B_1$; the remaining requirements ii), iii) and iv) of Definition 3.4 resp. follow from Lemma 3.3 i), ii) and iv) resp.

(only if) Since O_1 is a subnet of O_2, $B_1 \subseteq B_2$, $E_1 \subseteq E_2$ and O_1 is an occurrence net by Proposition 2.2. Furthermore, $Min(O_1) = Min(O_2)$ by Lemma 3.5; thus, O_1 and p_1 satisfy Definition 3.1 i), since O_2 and p_2 do and since p_1 is a restriction of p_2.

Since O_1 is a subnet of O_2 and contains with each $x \in B_2 \cup E_2$ also $pre(x)$ formed in O_2, this set is the same in O_1 and O_2. The same applies for ${}^\bullet x$ and for $\hat{e}$ and $e^\bullet$ if $e \in E_2$. With this, it is easy to see that p_1 is a homomorphism and that (O_1, p_1) is a b-process and canonical. □

Theorem 3.2 and Lemma 3.6 imply

Corollary 3.7 *Each prefix of a b-process is a b-process.*

Similarly to [Eng91, Theorem 23], we show:

Theorem 3.8 *The prefix-relation is a partial order for canonical b-processes of a net N. These form a complete lattice, i.e. each family of canonical b-processes has a least upper bound; this is simply their (componentwise) union – or the net graph with just the conditions $\{(s, \emptyset) \mid s \in M_N\}$ if the family is empty.*

Proof: The first statement is direct from Lemma 3.6. So let $(O_i, p_i)_{i \in I}$ be a non-empty family of canonical b-processes and (O, p) their union. (The case of the empty family is obvious.) It suffices to show that (O, p) is a canonical b-process.

By Lemma 3.3 i) and ii), each $x \in B \cup E$ has the same set $pre(x)$ in all (O_i, p_i) where it exists, and hence also in (O, p); the same applies to ${}^\bullet x$ and to $\hat{e}$ and $e^\bullet$ for $e \in E$ due to the labelling. Thus, causal closure and the co-set property are the same in all (O_i, p_i) and in (O, p), which implies that O is an occurrence net, p a homomorphism and (O, p) canonical. All the $Min(O_i)$ and $Min(O)$ coincide, and by canonicity $pre(e_1) = pre(e_2) \wedge p(e_1) = p(e_2) \Rightarrow e_1 = (p(e_1), pre(e_1)) = e_2$; hence, (O, p) is a canonical b-process. □

In particular, the canonical b-processes of a net N have a greatest element, called the *unfolding* $Unf(N)$ of N.

We now give a constructive characterization of the unfolding, compare [Eng91, Theorem 26]. Note that we will see below (Theorem 4.4) that the condition 'p is injective on X' can actually be omitted from the following theorem.

Theorem 3.9 *A canonical b-process (O,p) is the unfolding of N if and only if:*

(*) $\forall t \in T$, co-set $X \subseteq B$: if p is injective on X and $p(X) = pre(t)$, then there is $e \in E$ with $pre(e) = X$ and $p(e) = t$.

Proof: (only if) Assume t and X exist violating (*). Then we add the new event $e = (t, X)$, new conditions for $e^\bullet$ and the respective arcs and read arcs. This addition does not change $Min(O)$; also $pre(x)$ for $x \in B \cup E$, causal closures and co-sets in O stay as they are. It is easy to see that the construction gives a canonical b-process. Hence, (O,p) is a proper prefix of this new canonical b-process by Lemma 3.6 and not the unfolding.

(if) Assume (O,p) satisfies (*) and is a proper prefix of some canonical b-process (O',p'). Since $Min(O') = Min(O)$ by Lemma 3.5 and since $e^\bullet$ for $e \in E$ is 'saturated' in O, each new condition b in O' also has a new input event e with ${}^\bullet b = \{e\}$.

If there is a new event $e = (t, X)$ with $X \subseteq B$, then p' and thus p would be injective on X with $p(X) = pre(t)$. Hence, O has some $e' \in E$ with $pre(e') = X$ and $p(e') = t$ by (*) and thus (O', p') violates 3.1 ii).

We conclude that for any new event $e = (t, X)$ we could find a new condition $b \notin B$ with $b \in X = pre(e)$, which would have a new input event and so on. Hence, we could use edges in $F' \cup A'$ backwards from any new element of O' to construct an infinite backward chain of new elements; this either contradicts finitarity of O' or shows that, for some new event e, $pre(e)$ is not a co-set since $\downarrow pre(e)$ contains a contextual cycle. □

As in the ordinary setting without read arcs, this theorem shows how to 'construct' the unfolding: start with the conditions $\{(s, \emptyset) \mid s \in M_N\}$; then repeatedly choose t and X violating (*) and extend the canonical b-process as described in the only-if part of the proof above. This procedure usually runs forever, but it generates the correct result if each (t, X) is treated eventually. (Note that at each stage there are only finitely many (t, X).)

We now have to show that the unfolding contains the essential information about the net N. Then we can discuss how to apply a finite part of the above procedure to construct a finite prefix that already contains all this information.

4 Configurations, cuts and reachable markings

A set C of events of a b-process (O,p) of a net N is called *causally closed in E* if $\downarrow C \cap E = C$. For such a C, let $bp(C)$ be the subnet induced by the conditions $Min(O) \cup C^\bullet$ and the events C. (In fact, this is a prefix of (O,p); compare e.g. the next lemma.) C is a *configuration* if it is a finite set of events, which is causally closed in E, conflict-free and such that $bp(C)$ has no contextual cycle, i.e. $\sqsubset$ is a partial order for $bp(C)$. It is easy to see that $bp(C)$ is finite since C and N are finite.

For a configuration C, we define $Cut(C) = (Min(O) \cup C^\bullet) \setminus {}^\bullet C$ and $Mark(C) = p(Cut(C))$.

Lemma 4.1 *Let (O,p) be a b-process and C a configuration. Then $\downarrow Cut(C) = Min(O) \cup C^\bullet \cup C$, and this is the set of events and conditions of $bp(C)$.*

Proof: $Y = Min(O) \cup C^\bullet \cup C$ is causally closed, since for each $b \in pre(e)$, $e \in C$, we have $b \in Min(O)$ or C contains the unique event e' with $e'^\bullet = \{b\}$. Y contains $Cut(C)$, hence inclusion follows.

Now let $x \in Y$. We can construct a maximal path in Y with edges in $F \cup A$ starting in x, which ends in some y. By T-restrictedness and maximality, $y \notin C$. If $y \in Y \setminus C$, $y \notin {}^\bullet C$ by maximality, hence $y \in \mathit{Cut}(C)$ and $x \in \downarrow \mathit{Cut}(C)$. □

For a configuration C, $bp(C)$ is a process as defined e.g. in [Vog97, Section 4]; note that 'occurrence net' has a different meaning in that paper. $\mathit{Cut}(C)$ is $bp(C)^\bullet$ as defined there, i.e. the set of conditions with empty postset [Vog97, 4.2 vi)]; hence, p is injective on $\mathit{Cut}(C)$, and $\mathit{Mark}(C)$ is a reachable marking of N [Vog97, 4.5 and 4.8]. It is reached by firing the $p(e)$, $e \in C$, according to a linearization of $\sqsubset$. Also note that each event e of a b-process induces a *local configuration* $[e] = \{e' \mid e' \leq e\}$; obviously, $[e]$ is causally closed in E and it is a configuration since $pre(e)$ is a co-set. Thus each $p(e)$, $e \in E$, is a firable transition. We also regard the empty configuration as a local configuration.

Vice versa, each process as defined in [Vog97] is a b-process [2] of N, while its events form a configuration. Hence, by [Vog97, 4.10], each firable transition is represented as $p(e)$, e an event of the unfolding, and each reachable marking is some $\mathit{Mark}(C)$, C a configuration of the unfolding of N. We want to formulate the latter representation in terms of cuts; hence we show:

Lemma 4.2 *Let (O,p) be a b-process and C a configuration. Then $\mathit{Cut}(C)$ is a cut.*

Proof: Let $Y = \downarrow \mathit{Cut}(C)$. By Lemma 4.1 and definition of a configuration, Y is conflict-free and without contextual cycle. Furthermore, $\mathit{Cut}(C)$ may coexist since by definition $b^\bullet \cap C = \emptyset$ for all $b \in \mathit{Cut}(C)$. Thus, $\mathit{Cut}(C)$ is a co-set.

Assume $\mathit{Cut}(C) \subseteq X$ and X is a co-set with $x \in X$. Since X may coexist and $C \subseteq \downarrow X$, we have $x^\bullet \cap C = \emptyset$, i.e. $x \notin {}^\bullet C$.

If $x \in \mathit{Min}(O)$, then $x \in \mathit{Cut}(C)$; otherwise take e with ${}^\bullet x = \{e\}$. A maximal path with edges in F leading to e must start in $\mathit{Min}(O)$ since O is T-restricted and finitary; this path cannot leave $\downarrow \mathit{Cut}(C)$ immediately after an event e', since $e' \in C$ and $e'^\bullet \subseteq \downarrow \mathit{Cut}(C)$ by Lemma 4.1. If the path leaves $\downarrow \mathit{Cut}(C)$ immediately after a condition b, then either $b \in {}^\bullet C$ and we would have a conflict in $\downarrow X$ at b, or $b \notin {}^\bullet C$, i.e. $b \in \mathit{Cut}(C)$ and X may not coexist since $b^\bullet \cap \downarrow X \neq \emptyset$. Thus, the path stays in $\downarrow \mathit{Cut}(C)$, which implies $e \in C$ and $x \in C^\bullet$. Since $x \notin {}^\bullet C$, we get in this case, too, that $x \in \mathit{Cut}(C)$; hence, $\mathit{Cut}(C) = X$ and $\mathit{Cut}(C)$ is a cut. □

Lemma 4.3 *Let (O,p) be a b-process and X be a finite co-set. Then $C = \downarrow X \cap E$ is a configuration with $X \subseteq \mathit{Cut}(C)$.*

Proof: C is finite, since O is finitary, and causally closed in E since $\downarrow X$ is causally closed. Furthermore, $\downarrow X$ (hence C) is conflict-free and $bp(C)$ has no contextual cycle. ($bp(C)$ may have some additional conditions not in $\downarrow X$, which are not in ${}^\bullet C \cup \hat{C} \subseteq \downarrow C \subseteq \downarrow X$; hence, these have no outgoing ordinary or read arcs in $bp(C)$ and cannot be on a contextual cycle.) Thus, C is a configuration and, by definition of C, $X \subseteq \mathit{Min}(O) \cup C^\bullet$; furthermore, $X \cap {}^\bullet C = \emptyset$ since X may coexist, hence, $X \subseteq \mathit{Cut}(C)$. □

Theorem 4.4 *Let (O,p) be a b-process. Each co-set X is finite and p is injective on X. The cuts of (O,p) are exactly the sets $\mathit{Cut}(C)$, where C a configuration.*

[2] It should be noted that our unfolding is the overlay of the processes as defined in [JK95, MR95, Vog97]. Hence, relative to these references, our approach is 'right'.

Proof: Let X be a finite co-set, $C = {\downarrow} X \cap E$. Then C is a configuration with $X \subseteq \mathit{Cut}(C)$ by Lemma 4.3. By the considerations before Lemma 4.2, p is injective on $\mathit{Cut}(C)$ and X, hence $|X| \leq |S|$. Since subsets of a co-set are co-sets, this implies that no co-set can be infinite.

By Lemma 4.2, $\mathit{Cut}(C)$ is a cut if C is a configuration. Vice versa, let X be a cut and $C = {\downarrow} X \cap E$. By the first part of this proof, X is finite and C is a configuration with $X \subseteq \mathit{Cut}(C)$. Since X is a cut and $\mathit{Cut}(C)$ is a co-set (even a cut) this implies $X = \mathit{Cut}(C)$. □

From this theorem and the results from [Vog97] discussed before, we obtain:

Corollary 4.5 *Let (O,p) be the unfolding of N. Then p is injective on all cuts of (O,p). The reachable markings of N are exactly the sets $p(X)$ with X a cut (or the sets $\mathit{Mark}(C)$ with C a configuration); the firable transitions of N are exactly the $p(e)$, $e \in E$.*

The purpose of the finite prefix algorithm of McMillan is to generate a finite prefix of $\mathit{Unf}(N)$ that contains the full information on N in the sense of this corollary. For the correctness proof of this algorithm, some further results are needed.

If (O,p) is a b-process with a cut D, then the *suffix* $\Uparrow D = (O',p')$ consists of the subnet O' of O induced by Z and the restriction p' of p to Z, where $Z \subseteq B \cup E$ is the least set with $D \subseteq Z$ and $\forall x \in B \cup E : \mathit{pre}(x) \subseteq Z \Rightarrow x \in Z$.

Theorem 4.6 *Let (O,p) be a b-process of N and D a cut of (O,p). Then $\Uparrow D$ is a b-process of the net $N' = (S,T,W,R,p(D))$. If (O,p) is the unfolding of N, then $\Uparrow D$ is isomorphic to the unfolding of N'.*

Proof: Denote $\Uparrow D$ by (O',p') and $B' \cup E'$ by Z as above. We use $\downarrow$ for the causal closure formed in O; pre is also formed in O, while pre' is formed in O'. Note that

(**) for $z \in Z$, either $z \notin D$ and $\mathit{pre}(z) = \mathit{pre}'(z)$ or $z \in D$ and $\mathit{pre}'(z) = \emptyset$

– if we had $e \in \mathit{pre}'(z)$ and $z \in D$, we could by definition of Z find a path with edges in $F' \cup A'$ from some e' to e and then to z such that $\mathit{pre}'(e') \subseteq D$; by T-restrictedness this would show that z and some $z' \in {}^\bullet e'$ may not coexist.

Since O' is a subnet of O, it is an occurrence net by Proposition 2.2, and if $X \subseteq Z$ is a co-set in O, it is also one in O'. We also derive the converse: let $X \subseteq Z$ be a co-set in O', Y' be its causal closure in O' and $Y = {\downarrow} X$. We first study Y.

We have $Y \subseteq Y' \cup {\downarrow} D$: for each $y \in Y$, there is a path with edges in $F \cup A$ from y to some $x \in X \subseteq Y'$; whenever some z on this path is in $Y' \subseteq Z$ then by (**) $\mathit{pre}(z) = \mathit{pre}'(z) \subseteq Y'$, i.e. the preceding element is in Y', too, or $z \in D$ – in which case the path up to z stays in ${\downarrow} D$. We conclude that the path and in particular y is in $Y' \cup {\downarrow} D$.

Also note that $Y' \cap {\downarrow} D \subseteq D$: for all $x \in Y' \setminus D$, there is a path with edges in $F \cup A$ from some $d' \in D$ to x whose first event e' satisfies $\mathit{pre}'(e') \subseteq D$ by definition of Z, i.e. w.l.o.g. $d' \in {}^\bullet e'$ by T-restrictedness; for all $x \in {\downarrow} D$, there is a path with edges in $F \cup A$ from x to some $d \in D$; hence, if $x \in (Y' \cap {\downarrow} D) \setminus D$ exists, we can concatenate these paths; this shows $e' \in {\downarrow} \{d\}$ and D may not coexist due to d' and is not a cut.

Now we check the three conditions for X being a co-set in O:

- Events $e \in Y'$ have $\mathit{pre}(e) \subseteq Y'$ by (**), events $e \in {\downarrow} D$ have $\mathit{pre}(e) \subseteq {\downarrow} D$. Hence, if events $e_1, e_2 \in {\downarrow} X = Y \subseteq Y' \cup {\downarrow} D$ have ${}^\bullet e_1 \cap {}^\bullet e_2 \neq \emptyset$, then $e_1, e_2 \in Y'$ and $e_1 = e_2$

since Y' is conflict-free, or $e_1, e_2 \in \downarrow D$ and $e_1 = e_2$ since $\downarrow D$ is conflict-free, or $e_1 \in Y'$ and $e_2 \in \downarrow D$ and ${}^\bullet e_1 \cap {}^\bullet e_2 \subseteq Y' \cap \downarrow D \subseteq D$ (or vice versa). But the last clause is impossible since e_2 belongs to the configuration C with $Cut(C) = D$ by Lemma 4.3, where $D \cap {}^\bullet C = \emptyset$.

- Assume a contextual cycle in $Y = \downarrow X$. The cycle cannot only have events in $C = \downarrow D \cap E$, since in this case it would be in $bp(C)$ and D would not be a co-set. Also, the cycle cannot only have events in Y' since X is a co-set in O'. Thus, there must be some $e' \in Y' \cap E$ on the cycle where the next event e is in C. This implies $e'(F \circ F \cup F \circ A \cup A^{-1} \circ F)e$; $e'(F \circ F \cup F \circ A)e$ would give $e' \in Y' \cap \downarrow D$ but $e' \notin D$ (since e' is an event), a contradiction. So we have a condition b with $e'A^{-1}bFe$, hence $b \in {}^\bullet C \subseteq \downarrow D \setminus D$ and $b \in pre(e') \subseteq Y'$ (recall (**)), contradicting again $Y' \cap \downarrow D \subseteq D$.

- Assume $b \in X$, $e \in Y = \downarrow X$ and $b \in {}^\bullet e$. Then either $e \in Y'$ and X cannot coexist in O' or $e \in \downarrow D$, $b \in Y' \cap \downarrow D \subseteq D$ and D cannot coexist in O.

So far, we have seen that O' is an occurrence net and that

(***) $X \subseteq Z$ is a co-set in O' if and only if it is a co-set in O.

Whenever we have included an event e in Z, then $pre(e) \subseteq Z$ and $e^\bullet \subseteq Z$, since for $b \in e^\bullet$, $pre(b) = \{e\} \subseteq Z$. Thus, p' is a homomorphism. As noted above, $pre'(d) = \emptyset$ for all $d \in D$, and $Min(O') = D$ by definition of $\Uparrow D$; since D is a cut, p and hence p' is injective on D. Furthermore, $pre(e)$ is the same in O and O' for $e \in E'$; hence, (O', p') is a b-process as required.

Finally, if (O, p) is the unfolding of N, it satisfies (*) in Theorem 3.9. Then, (O', p') also satisfies (*) by (***) and is isomorphic to the unfolding of N' by Theorem 3.9 (and 3.2). □

5 The finite prefix algorithm and read-persistent nets

In this section we apply the (improved) McMillan's algorithm [ERV96] for constructing a finite prefix of the unfolding. For this we restrict ourselves to a subclass of Petri nets with read arcs, in such a way that (basically) we do not allow a net to have a marking where two transitions with a common precondition s are enabled whilst one of them uses s as a preplace and the other as a read place. We will call this class read-persistent nets. We will also show a counter-example preventing the application of the standard cutoff condition to a net which is not read-persistent.

This restriction may be look severe from the viewpoint of the rationale behind the entire class of Petri nets with read arcs. Indeed, our examples in Introduction (cf. Figure 3 and 4) show that the place replication strategy, which helps mitigate explosion of reading transitions in unfolding an ordinary Petri net with loops, cannot avoid combinatorial blow-up for transitions which consume tokens from a read place – but the latter situation cannot happen in read-persistent nets. So, two questions arise. Firstly, does the new unfolding with read arcs bring any size or timing savings for this subclass, or can we simply use the loops-replication technique and the existing unfolding? Secondly, is this subclass practically useful? Fortunately, at the end of this section we will be able to answer these questions positively. Furthermore, the fact that the loops-replication strategy alters the original net makes the interpretation of the analysis results in terms of the original net a problem.

input A safe Petri net with read arcs $N = (S, T, W, R, M_N)$
output A complete finite prefix (O_f, p_f) of unfolding $Unf(N) = (O, p)$ and a set of cutoff events E_c
begin
  Initialise $O_f = (B, E, F, A)$ and p_f with conditions $\{b : p_f(b) = s \in M_N\}$
  Initialise E_c as empty
  Initialise Q (a queue of enabled transitions with preset) with
  all $(t, \emptyset)$ where t is enabled at M_N ($pre(t) \subseteq M_N$)
  while Q is not empty **do**
    Pull (t, X) from Q
    Add to O_f new e with $p_f(e) = t$ and $pre(e) = X$,
    new $\{b : p_f(b) \in t^\bullet\}$ and new arcs
(*) **if** *e is a cutoff* **then do**
      Add e to E_c and remove [a] all $s \in e^\bullet$
     enddo
    forall new pairs (t, X) where t in T and
(**) $X \subset B$ *is a co-set with* $pre(t) = p_f(X)$ **do**
      Add (t, X) to Q in order of $|\downarrow X \cap E|$ (size of
      local configuration of instance of t)
    enddo
  enddo
  return (O_f, p_f) and E_c
end

[a] We remove the postconditions of a cutoff event in order to prevent further unfolding. Technically, the result is not a prefix, not even a b-process because it is not T-restricted (cf. Section 2).

Figure 6 The finite prefix algorithm.

Let us first consider the finite prefix construction algorithm shown in Figure 6.

The execution of the line marked with (*) calls for a separate function which decides whether a newly created event is a cutoff event. McMillan's algorithm in [McM93] uses the following condition, which is applicable to any type of net: e is a *cutoff* if $\exists e' \in E$: $Mark([e']) = Mark([e]) \wedge ||[e']|| < ||[e]||$ or if $Mark([e]) = M_N$. Note that in the above situation, the algorithm in Figure 6, always generates event e' *before* event e due to the ordering in queue Q. McMillan's cutoff condition is based on a partial ordering of local configurations; it can lead to redundancy (multiple representation of the same marking in the unfolding prefix) when $||[e']|| = ||[e]||$. An improved total ordering for safe nets is given in [ERV96], which can also be applied here.

The line marked with (**) in this algorithm has a specific meaning for Petri nets with read arcs (cf. definition of co-set in Section 2). There is however a problem with using this algorithm for arbitrary nets with read arcs. Consider the example of a net shown in Figure 7(a) and its unfolding prefix in Figure 7(b). The prefix event labelled c is a cutoff because $Mark([c]) = M_N = \{1, 4\}$. This prefix, generated by the algorithm in Figure 6, is however incomplete – it does not cover markings $\{1, 5\}$ and $\{2, 5\}$.

The problem is that the net in our example is not read-persistent – event c is consuming

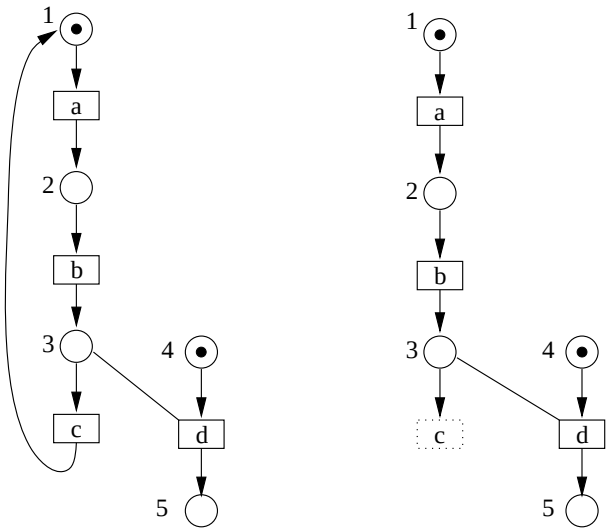

Figure 7 Example illustrating the problem with a net that is not read-persistent; (a) original net with read arc, between place 3 and transition d; (b) prefix generated by algorithm in Figure 6.

from place 3 whilst event d is reading this place. Lemma 5.2 below fails when applied to such nets. One could say that the same event c has two local configurations (where one would use a different definition from ours), $C_1 = \{a, b, c\}$ and $C_2 = \{a, b, d, c\}$, which are overlaid in the same occurrence net. For C_1 event c is cutoff but not for C_2. A possible solution would be to continue unfolding beyond c, thus requiring that a new cutoff condition checks whether an event is a cutoff for all its 'alternative local configurations'. Investigating this condition is however left outside the scope of this paper, which restricts consideration to the class of read-persistent nets.

Our main argument for the practicality of read-persistent nets is that they exactly correspond to the behaviour of hazard-free digital circuits. Indeed, let us assume the circuit model described in Introduction. Let a read arc enable a transition which represents an output of a gate and let a consuming arc from the same place lead to a transition which corresponds to the gate's input. The fact that there is a marking when both such transitions are enabled manifests a potential hazard – the output signal transition can fire or be disabled (non-deterministically) by the input signal transition. Due to this analogy, the problem of checking hazard-freedom in an asynchronous circuit can be posed as that of verifying whether a corresponding net with read arcs is read-persistent.

Before incorporating this extra condition in our prefix algorithm we need the following background.

Let N be a net, M a marking. Then $t_1, t_2 \in T$ are *in conflict under* M, if $M[t_1\rangle$, $M[t_2\rangle$ and ${}^\bullet t_1 \cap pre(t_2) \neq \emptyset$ or ${}^\bullet t_2 \cap pre(t_1) \neq \emptyset$, i.e. if both, t_1 and t_2, are enabled but one consumes a token that is also needed by the other. N is *read-persistent*, if ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$ for all $t_1, t_2 \in T$ in conflict under some reachable marking; i.e. the transitions are not only in conflict because one wants to check (read) a token that the other wants to consume (write).

The following theorem shows that the situation with read-persistent nets is much simpler than with general nets with read arcs; in particular, when checking whether a set X in a b-process is a co-set, we do not have to check for contextual cycles. Even if the savings by read arcs are limited in the case of read-persistent nets, this shows that these savings come with no price to pay since the computation of a finite prefix is just as efficient for

read-persistent nets with read arcs as for ordinary nets.

Theorem 5.1 *Let N be read-persistent and Y be a causally closed conflict-free set in $Unf(N)$. Then $<$ and $\sqsubset_Y$ coincide on Y. If X is a set in $Unf(N)$ such that $\downarrow X$ is conflict-free and X may coexist, then X is a co-set.*

Proof: We first note that a set Y where $<$ and $\sqsubset_Y$ coincide cannot have a contextual cycle, since $<$ is a partial order. In particular, this shows that the second claim is implied by the first one.

Let $Unf(N) = (O, p)$. We proceed by induction on $|Y|$, so let Y be a minimal counter-example, i.e. there are $e_1, e_2 \in Y$ with $e_1 A^{-1} b F e_2$ but not $e_1 < e_2$. We can neither have $e_2 < e_1$, since then $\downarrow pre(e_1)$ would have a contextual cycle through e_1 and e_2, thus $pre(e_1)$ would not be a co-set.

We have $Y = \downarrow \{e_1, e_2\}$ by minimality of Y. ($\downarrow \{e_1, e_2\}$ is a counter-example, too.) Furthermore, Y is the disjoint union of $\{e_1, e_2\}$ and $Y' = \downarrow (pre(e_1) \cup pre(e_2))$. We show that $pre(e_1) \cup pre(e_2)$ is a co-set.

Y' is conflict-free, since Y is. Y' is smaller than Y, hence $<$ and $\sqsubset_{Y'}$ coincide on Y' and, in particular, Y' does not contain a contextual cycle. If $pre(e_1) \cup pre(e_2)$ may not coexist, then there is some $b' \in pre(e_i)$ with $i \in \{1, 2\}$ and $e \in Y'$ with $b' F e$. If $b' F e_i$, Y would not be conflict-free; hence $b' A e_i$. $Y' \cup \{e_i\}$ is conflict-free, causally closed and smaller than Y, hence $<$ and $\sqsubset$ coincide on it; thus, $e_i (A^{-1} \circ F) e$ shows $e_i < e$. But $e \in \downarrow \{e_1, e_2\}$, i.e. $e < e_1$ or $e < e_2$. In one case, we get $e_i < e_i$, in the other $e_i < e_{3-i}$, in any case a contradiction.

Thus, $pre(e_1) \cup pre(e_2)$ is a co-set which can be extended to a cut D, such that p is injective on D and $M = p(D)$ is a reachable marking of N. We have $M[p(e_1)\rangle$ and $M[p(e_2)\rangle$ and $e_1 A^{-1} b F e_2$ shows that $p(e_1)$ and $p(e_2)$ are in conflict under M. Thus, by read/write persistence, there is some $s \in {}^\bullet p(e_1) \cap {}^\bullet p(e_2)$, a unique $c \in pre(e_1) \cup pre(e_2)$ with $p(c) = s$ and $c \in {}^\bullet e_1 \cap {}^\bullet e_2$. Hence, Y is not conflict-free, a contradiction. □

There are examples of read-persistent nets where $<$ and $\sqsubset$ do not coincide globally. For instance, let s be a preplace for an event a and a read place for an event b. Let a and b also be in conflict, due to some other preceding place, which helps the net not violate read persistence for a and b, because they cannot be enabled simultaneously, under the same marking. Then, globally, we have $b \sqsubset a$ (effect of 'conditional' nature of $\sqsubset$ since they cannot happen in the same configuration) but not $b < a$.

We now come to the result that is needed to prove the correctness of the finite prefix algorithm, but fails in the case of general nets with read arcs.

If $C \subseteq C'$ are configurations of $Unf(N)$ for some net N, then we call $C' \setminus C$ a *tail* and C' an *extension* of C.

Lemma 5.2 *Let N be a read-persistent net, D a cut of $Unf(N)$ and $C = \downarrow D \cap E$. Then the tails of C are the configurations of $\Uparrow D$.*

Proof: "$\Leftarrow$": Let C' be a configuration of $\Uparrow D$, $X = (D \cup C'^\bullet) \setminus {}^\bullet C'$ the corresponding cut in $\Uparrow D$, Y' the causal closure of X in $\Uparrow D$ and $Y = \downarrow X$, where $\downarrow$ is always formed in $Unf(N)$.

Since X is a cut in $\Uparrow D$, we have $D = Min(\Uparrow D) \subseteq Y' \subseteq Y$; thus, $Y' \cup \downarrow D \subseteq Y$ and $D \subseteq Y' \cap \downarrow D$. As in the proof of Theorem 4.6, we get $Y = Y' \cup \downarrow D$ and $Y' \cap \downarrow D = D$. Since $C' = Y' \cap E$, we get that $C' \cup C$ is the configuration $Y \cap E$ and C' is the respective tail of C.

"$\Rightarrow$": Let $C \cup C'$ be a configuration of $Unf(N)$ with $C \cap C' = \emptyset$.

(*) $\forall e \in C' : pre(e) \cap {}^\bullet C = \emptyset$

Proof of (*): Otherwise, we have $e \in C'$, $e_0 \in C$ and some $b \in pre(e) \cap {}^\bullet e_0$. Since $C \cup C'$ is conflict-free, this implies $b \in \hat{e} \cap {}^\bullet e_0$, i.e. $e(A^{-1} \circ F)e_0$. By Theorem 5.1, this implies $e < e_0$, hence $e \in \downarrow C$ and C is not a configuration. $\Box(*)$

Now let $e_1, \ldots, e_n$ be a linearization of $(C', <)$. $C_i = C \cup \{e_1, \ldots, e_i\}$ is a configuration, hence $pre(e_i) \subseteq Min(O) \cup C_{i-1}^\bullet$. By (*), $pre(e_i) \subseteq (Min(O) \cup C^\bullet) \setminus {}^\bullet C \cup \{e_1, \ldots, e_{i-1}\}^\bullet = D \cup \{e_1, \ldots, e_{i-1}\}^\bullet$. Now with induction, $pre(e_i)$, e_i and $D \cup \{e_1, \ldots, e_i\}^\bullet$ are in $\Uparrow D$ and C' is a configuration in $\Uparrow D$. $\Box$

The essential application of this lemma is roughly the following: if we declare e a cut-off event, we will not generate the extensions of $[e]$, so we could miss the corresponding markings. But these extensions correspond to tails of $[e]$, hence configurations of $\Uparrow Cut([e])$. These are by Theorem 4.6 essentially the configurations of some $\Uparrow Cut([e'])$, where $Mark([e]) = Mark([e'])$ and the event e' of the finite prefix is not a cut-off event; thus, we will find the above markings by considering extensions of $[e']$.

Now we can partially order configurations C_1, C_2 by $|C_1| < |C_2|$. This partial order satisfies the requirements of [ERV96, Definition 3.3]; in particular, the third requirement follows by a slight generalization of the argument above. Following the proof in [ERV96] we conclude:

Theorem 5.3 *Let N be a read-persistent net. Then the finite prefix algorithm terminates with a finite b-process (O, p). A marking M of N is reachable if and only if there is a cut D in (O, p) with $p(D) = M$. A transition t is firable under a reachable marking of N if and only if there is an event e in O with $p(e) = t$.*

The finite prefix algorithm can check on-line whether the given net is read-persistent. We simply need to add the following condition to be checked before adding a new event e into the prefix:

$$\exists e' \in E, Y \subset E : ({}^\bullet e \cap \hat{e}' \neq \emptyset \vee {}^\bullet e' \cap \hat{e} \neq \emptyset) \wedge {}^\bullet e \cap {}^\bullet e' = \emptyset \wedge X \cup X' \subseteq Y,$$

where $X = pre(e), X' = pre(e')$ and Y is a co-set. If this condition is true, the algorithm stops and the net is reported not read-persistent.

The above theorem holds also in the case when the algorithm uses the cutoff condition according to the improvement of [ERV96] (to the original McMillan's cutoff condition). The reasons for this are quite obvious [3]. In the case of read-persistent nets we cannot override the order in which transitions are fired, and hence events are instantiated in the unfolding. The 'help' provided by the lexicographic part of the improved order between configurations imposed by McMillan's size-based partial order (in which events are pulled out of the queue in the algorithm) is not effected by the places (nor effects the instantiation thereof) that are read by events. The causality order, which governs precedence in read-persistent nets, where each event has exactly *one* local configuration, is not different from the causality order in ordinary Petri nets. Therefore the "lexicographic signatures of configurations", which determine the instantiation order in the cases where two transitions, candidates for one of them being a cutoff, have the same size of their local configurations, also remain unique.

[3] We avoid here additional formal constructions, such as definition of the total order $\prec_r$ between configurations, which is a combination of the partial order induced by the size of configurations and the lexicographic order of the configurations' elements. An interested reader is referred to [ERV96] for these details.

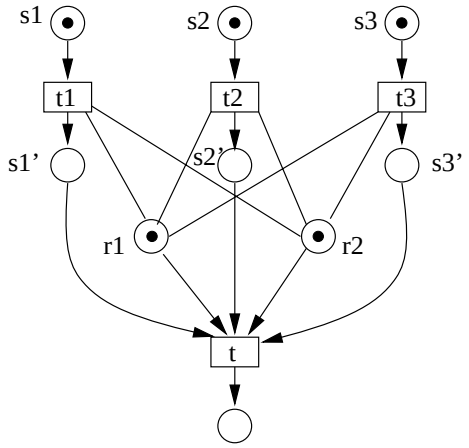

Figure 8 Example illustrating savings with read arcs compared to loops-replication strategy (for read-persistent nets)

We conclude this section with an illustration of the fact that even for the class of read-persistent nets the construction of a prefix based on the unfolding with read arcs provides considerable savings in comparison with the use of loops-replication strategy.

Consider a net with n transitions $t_1, \ldots, t_n$, reading a set of m places $r_1, \ldots, r_m$ concurrently as shown in Figure 8 for $n = 3$ and $m = 2$. An additional transition t consumes tokens from all n postplaces of the above-mentioned transitions and from all m above-mentioned read places. This net is (up to the marking) its own unfolding with read arcs. Note that it is a read-persistent – the consuming transition t is enabled only after all reading transitions have fired.

The size of this unfolding in terms of the total number of places and transitions is in general $3n + m + 2$; together with the arcs it is $6n + nm + 2m + 2$.

If we use loops-replication technique, the new net will have nm loop places. The size of the new (ordinary net) unfolding will become $3n + nm + 2$ places and transitions and overall $6n + 5nm + 2$. For the case $n = m$, e.g., the savings with the read arcs are almost fivefold.

Furthermore, in the second version, each loop place is represented twice in the unfolding, and hence there are 2^n co-sets (just consisting of the loop places) to consider when trying to instantiate transition t; all this is a time overhead for the prefix algorithm. (It is of course partly balanced by overheads involved in checking read-persistence conditions for the version with read arcs, but we assume that this check has to be performed anyway.

Finally, let us remove tokens from all places s_i but s_1. The unfolding with read arcs will have just one transition with $m + 2$ places (and as many arcs). The unfolding with replication gives instead $nm + 2$ places.

6 Conclusion

Petri nets with read arcs find application in a number of areas, such as modelling and verification of concurrent programs and asynchronous circuits. Apart from being more adequate from the purely semantical point of view (ordinary nets have destructive-read-and-rewrite semantics), they can help more efficient model checking when using their unfoldings. In this paper we have defined the main theoretical elements for the construction of a finite prefix of the unfolding of a safe Petri net with read arcs. Those include the notions of a conditional

precedence relation, a contextual cycle, and a co-set. The latter has a specific feature when compared to that of ordinary nets: it cannot be generalised naturally from the binary concurrency relation. Based on the new notion of a co-set, we have been able to redefine the concepts of occurrence net, branching process and unfolding.

We have demonstrated that the existing algorithms for prefix construction, cf. [McM93, ERV96], cannot be applied directly to nets with read arcs in general; possibly, one would have to consider multiple local configurations for events and take them into account in the definition of a cutoff event. But we have shown that for a subclass called read-persistent nets direct application of the existing prefix algorithms is possible (with the appropriate change of the co-set condition). The class of read-persistent nets is however not too restrictive to render them impractical. The property of read persistence exactly corresponds to the notion of hazard-freedom often used as a synonym of asynchronous circuit correctness. Since it is fairly easy to check read-persistence on-line, whilst constructing the finite prefix, the proposed algorithm is effectively a way of verifying hazard-freedom.

The present work has therefore two main contributions. One is the theoretical framework for the branching processes and unfolding of nets with read arcs. The other is the algorithm for constructing a finite unfolding prefix for a practically useful subclass of nets with read arcs.

We have shown by examples (in Introduction) that the greatest savings in terms of the prefix size can be achieved when the net with read arcs is of general type, i.e. not necessarily read-persistent. Unfortunately, the use of the existing cutoff techniques [McM93, ERV96] is generally impossible for non-read-persistent nets (cf. the example in Figure 7). Developing a new condition for unfolding truncation, which works in general, is a subject of our current research.

References

[BG95] N. Busi and R. Gorrieri. A Petri net semantics for π-calculus. In L. Insup and S. Smolka, editors, *CONCUR 95*, Lect. Notes Comp. Sci. 962, 145–159. Springer, 1995.

[BP96] N. Busi and M. Pinna. Non-sequential semantics for contextual P/T-nets. In J. Billington and W. Reisig, editors, *Applications and Theory of Petri Nets 1996*, Lect. Notes Comp. Sci. 1091, 113–132. Springer, 1996.

[CH93] S. Christensen and N.D. Hansen. Coloured Petri nets extended with place capacities, test arcs, and inhibitor arcs. In M. Ajmone-Marsan, editor, *Applications and Theory of Petri Nets 1993*, Lect. Notes Comp. Sci. 691, 186–205. Springer, 1993.

[Eng91] J. Engelfriet. Branching processes of Petri nets. *Acta Informatica*, 28:575–591, 1991.

[EB96] J. Esparza and G. Bruns. Trapping mutual exclusion in the box calculus. *Theor. Comput. Sci.*, 153:95–128, 1996.

[ERV96] J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. In T. Margaria and B. Steffen, editors, *TACAS 96*, Lect. Notes Comp. Sci. 1055, 87–106. Springer, 1996.

[JK95] R. Janicki and M. Koutny. Semantics of inhibitor nets. *Information and Computation*, 123:1–16, 1995.

[McM93] K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Boston, 1993.

[MR95] U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32:545–596, 1995.

[NPW81] M. Nielsen, G.D. Plotkin, and G. Winskel. Petri nets, event structures and domains I. *Theor. Comput. Sci.*, 13:85–108, 1981.

[Vog91] W. Vogler. Executions: A New Partial Order Semantics of Petri Nets. . *Theor. Comput. Sci.*, 91:205-238, 1991.

[Vog97] W. Vogler. Partial order semantics and read arcs. Technical Report 1997-1, Inst. f. Informatik, Univ. Augsburg, 1997. See http://www.math.uni-augsburg.de/~vogler/; extended abstract in MFCS 97, LNCS 1295, 508–517.

[YKSK96] A. Yakovlev, A.M. Koelmans, A. Semenov and D.J. Kinniment. Modelling, analysis and synthesis of asynchronous control circuits using Petri nets. *INTEGRATION: the VLSI Journal*, 21:143-170, 1996.