Fernandes J, Koutny M, Mikulski L,
Pietkiewicz-Koutny M, Sokolov S, Yakovlev A.
Persistent and Non-violent Steps and the Design of GALS Systems.
***Fundamenta Informaticae* 2015, 137(1), 143-170.**

Fundamenta Informaticae XX (2014) 1–28
IOS Press

Persistent and Nonviolent Steps and the Design of GALS Systems

Johnson Fernandes
School of Electrical & Electronic Eng., Newcastle University, Newcastle upon Tyne, NE1 7RU, U.K.

Maciej Koutny
School of Computing Science, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.

Łukasz Mikulski
Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, Chopina 12/18, Poland

Marta Pietkiewicz-Koutny
School of Computing Science, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.

Danil Sokolov
School of Electrical & Electronic Eng., Newcastle University, Newcastle upon Tyne, NE1 7RU, U.K.

Alex Yakovlev
School of Electrical & Electronic Eng., Newcastle University, Newcastle upon Tyne, NE1 7RU, U.K.

Abstract. A concurrent system is persistent if throughout its operation no activity which became enabled can subsequently be prevented from being executed by any other activity. This is often a highly desirable (or even necessary) property; in particular, if the system is to be implemented in hardware. Over the past 40 years, persistence has been investigated and applied in practical implementations assuming that each activity is a single atomic action which can be represented, for example, by a single transition of a Petri net.

In this paper we investigate the behaviour of GALS (Globally Asynchronous Locally Synchronous) systems in the context of VLSI circuits. The specification of a system is given in the form of a Petri net. Our aim is to re-design the system to optimise signal management, by grouping together concurrent events. Looking at the concurrent reachability graph of the given Petri net, we are interested in discovering events that appear in 'bundles', so that they all can be executed in a single clock tick. The best candidates for bundles are sets of events that appear and re-appear over and over again in the same configurations, forming 'persistent' sets of events. Persistence was considered so far only in the context of sequential semantics. In this paper, we move to the realm of step based execution and consider not only steps which are persistent and cannot be disabled by other steps, but also steps which are nonviolent and cannot disable other steps. We then introduce a formal definition of a bundle and propose an algorithm to prune the behaviour of a system, so that only bundled steps remain. The pruned reachability graph represents the behaviour of a re-engineered system, which in turn can be implemented in a new Petri net using the standard techniques of net synthesis. The proposed algorithm prunes reachability graphs of persistent and safe nets leaving bundles that represent maximally concurrent steps.

Keywords: asynchronous and synchronous circuit, GALS system, persistence, nonviolence, step transition system, step semantics, Petri net.

1. Introduction

A concurrent system is persistent [2, 3, 4, 16] if throughout its operation no activity which became enabled can subsequently be prevented from being executed by any other activity. This is often a highly desirable (or even necessary) property; in particular, if the system is to be implemented in hardware [7, 8, 22]. Over the past 40 years, persistence has been investigated and applied in practical implementations assuming that each activity is a single atomic action which can be represented, for example, by a single transition of a Petri net (used as a formal model of a concurrent system). In other words, persistence was considered assuming the sequential execution semantics of concurrent systems [19].

Traditional circuit design styles have been following one of the two main strands, namely synchronous and asynchronous. In a nutshell, these two approaches differ in their techniques of synchronising interaction between circuit elements. Asynchronous designs adopt 'on request' synchronisation where interaction is regulated by means of handshake control signals. They are designed to be adaptive to delays of signal propagation. Synchronous designs, on the other hand, assume worst case delay of the critical path and determine a global periodic control signal for synchronisation called the *clock*. The clock signal limits the many sequencing options considered in asynchronous control. Thus synchronous circuits can be considered as a proper subset of asynchronous circuits [6].

Asynchronous logic was the dominant design style with most early computers. In particular, David Muller's speed-independent circuits, dating back to the late 1950s, have served many interesting applications such as the ILLIAC I and ILLIAC II computers [18]. However, since 1960, an era when fabrication of integrated circuits (ICs) became a feasible business, synchronous design became the mainstream technique as it met the market needs with its shorter design cycle. Today, majority of designs are synchronous, well etched in the heart of semiconductor industry together with superior CAD tools and EDA flows.

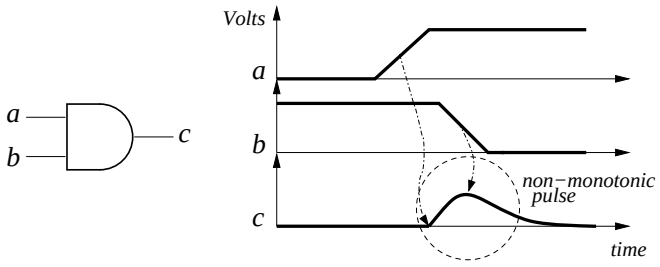


Figure 1: Hazardous switching of an AND gate.

One of the main issues with the complexity of asynchronous circuits was the handling of hazards. Hazards are manifestations of undesirable switching activity called glitches. In the asynchronous style of synchronisation, the output of each circuit element is potentially sensitive to its inputs. This can give rise to non-monotonic pulses (or glitches) when transitioning between output states, as illustrated in the waveform of Figure 1 taking the case of an AND gate. Due to tight timing between the rising edge of input a and falling edge of input b, the output c produces a non-monotonic pulse before stabilising to a low. This behaviour is hazardous as it is uncertain how the fanout of the AND gate will interpret the glitch; the output c temporary switching to logical 1 or staying at logical 0 all the time.

As shown, for instance, in paper [22], the phenomenon described in the above example can be conveniently interpreted in terms of formal models such as Keller's named transition systems [13] or Petri nets [6]. In particular, what we see in this circuit is the effect where a signal that is enabled (rising edge of c) in a certain state of the circuit may become disabled without being executed after the occurrence of another signal (falling edge of b). Such an effect corresponds to the violation of *persistence* property at the level of signal transitions if the latter are used to label the corresponding named transition system. Furthermore, when such a circuit is modelled by a labelled Petri net following the technique of [22], the Petri net would also be classified as a non-persistent one. Thus, it was shown in [22] that the modelling and analysis of an asynchronous circuit with respect to hazard-freedom is effectively reduced to the analysis of persistence of its corresponding Petri net model.

Synchronous circuits, on the other hand, do not require persistence satisfaction as they are intrinsically immune to hazardous behaviour. The principle reason being that the clock, set at worst-case latency period, filters out undesirable circuit switching. This greatly simplifies circuit design compared to asynchronous methods wherein the same circuit had to be analysed for persistence and redesigned to ensure glitch-free operation. Clocked circuits are thus preferred over asynchronous circuits for designing functionally correct (hazard-immune) ICs efficiently. However with chip sizes scaling to deep sub-micron level, semiconductors are experiencing severe variability and it is becoming extremely complicated to design chips in the synchronous fashion. This is because designing for variability requires longer safety margins which in turn reduces the clock frequency and degrades circuit performance. To cope with these challenges, asynchronous design methodologies have re-emerged owing to their *inherent* adaptiveness [9]. However, they still suffer significant challenges such as complicated design flow, high overhead costs from control and, lack of CAD support tools and legacy design reuse. Therefore attempts are being made to find a compromise.

An on-trend intermediate solution is mixed synchronous-asynchronous design, chiefly acting in the form of Globally Asynchronous Locally Synchronous (GALS) methodology. Its benefits are well known in literature [11, 12, 20]. GALS system design, introduced in [5], can exploit the advantages of asynchrony and at the same time maximally reuse the products of synchronous design flow. This design technique divides a digital system into synchronous islands which communicate asynchronously by handshake mechanism. Each island has its own local clock which can be activated on demand by means of a handshake control signal. Such systems comprise a mixed temporal behaviour. Asynchronous handshakes handle interaction between components where adaptability can significantly improve performance, while clocking is applied to components where worst case performance is tolerable. However, it is worth noting that modelling GALS systems would involve detection of potential hazardous states due to presence of asynchronous components, making their design and verification a significant research challenge.

Being a recent trend, there is a lack of formal models that describe correctness of GALS designs. The complexity in modelling them begins with the investigation of persistence. It should be noted that the standard notion of persistence has been defined at the level of single actions, which is also known as interleaving semantics of concurrency. This notion has been adequate for representing the correctness of the behaviour of circuits that are fully asynchronous. In asynchronous circuits, there is concurrency between independent actions and sequential order between causally related actions. This notion is well represented by Keller's named transition systems [13]. Figure 2(a) depicts such a model capturing the asynchronous behaviour of a system with four events: a, b, c and d. Now, in synchronous circuits, the clock signal would trigger a single action or several actions. These circuits exhibit parallelism between

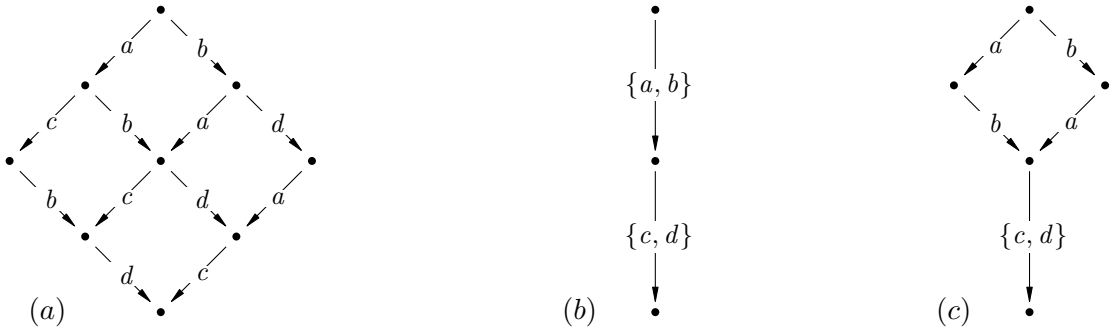


Figure 2: Temporal representations of systems having concurrent, parallel and mixed concurrent-parallel behaviours: (a) interleaving model for asynchronous behaviour; (b) step model for synchronous behaviour; and (c) mixed model for GALS behaviour.

actions in the same clock cycle and sequential order between groups of actions in adjacent clock cycles. To represent this group execution of actions, we will use *steps*, and therefore we need step transition systems to represent such a behaviour. A step represents a single action or a group of actions that are triggered simultaneously from a particular state by the clock signal. Figure 2(b) shows such a transition system model capturing the temporal behaviour of a synchronous system with the help of steps. For the case of GALS, there is a mixture of synchrony and asynchrony and hence both concurrent and parallel behaviour have to be represented. Figure 2(c) illustrates the mixed temporal behaviour seen in such systems. In all three cases, step transition systems provide a suitable behavioral model, as a single transition can be treated as a singleton step.

Synchronous and asynchronous systems have distinct techniques to guarantee functionally correct behaviour. However, for GALS systems, it is not so straightforward as correctness should be accounted from both angles. We would like to find an adequate representation of the correct behaviour of GALS systems. Here, it would be natural to define such a behaviour in analogous way as it was done for asynchronous circuits, i.e., with the use of the notion of persistence. However, when modelling GALS systems we have to consider complex actions, namely steps, and corresponding transition systems. This paper is hence centred around extending the notion of persistence to steps.

The main motivation for studying persistent steps in this paper is as follows. Digital system design based on formal models is normally associated with two main tasks: one is the verification of a system's behavioural specification or checking the model of the system implementation, while the other is the synthesis of the circuit implementation from its specification. In the context of verification we would like, for example, to check if the Petri net model of a GALS system satisfies the requirement of hazard-freedom under a particular form of synchronisation of actions (in steps). In the context of synthesis, we would like to find the optimal partitioning of actions into synchronous steps so that the complexity of control of these steps is minimised. For example, the intuitive complexity of handling synchronisations safely in the three scenarios of Figure 2 varies between them, from the most intricate in the fully asynchronous one (case (a)) to the simplest in fully synchronous one (case (b)), placing the GALS version in the middle (case (c)). With this varying complexity, one can design systems that may exhibit hazards if they are treated as fully asynchronous, but when actions are synchronised into steps the system would behave safely. Amongst the methods for synchronising actions into steps, we can consider those that are

based on the insertion of additional control circuits to physically 'bundle' actions together, or based on ensuring the appropriate 'bundling' constraints based on timing, or delays. Traditional globally clocked systems, self-timed systems working under fundamental mode assumptions, and asynchronous systems with relative timing [6] are all of the latter category.

It is this idea of bundling those steps of actions that are 'hazard-free' or persistent that motivated our notion of *bundles*, introduced in this paper. In terms of nets and corresponding transition systems, bundles are informally sets of transitions that can be executed synchronously and therefore be treated as some kind of 'atomic actions', giving rise to new 'bigger' transitions. Section 5 provides a more formal treatment for bundles and shows a constructive procedure for deriving them by pruning reachability graphs or transition systems, depending on whether we are solving the verification or synthesis problem. For example, in the process of synthesis of the control policy for a GALS system, such a 'pruned' transition system would represent the desired behaviour, which then we would like to implement in a form of a Petri net.

The paper is organised as follows. Section 2 recalls the basic definitions and notations concerning step transition systems and PT-nets. Section 3 introduces various types of persistent and nonviolent steps of transitions in PT-nets and Section 4 discusses their basic properties. Section 5 presents the main application result of the paper, an algorithm that prunes the concurrent reachability graph of a net, which serves as an initial system specification, to obtain a representation of a desired 'GALS' behavior. Finally, Section 6 contains conclusions and presents directions for future work.

2. Preliminaries

In this section we recall definitions and notations concerned with step transition systems and step semantics of Petri nets used throughout the paper.

Let T be a finite set of net transitions representing actions of a concurrent system. A set of transitions is called a *step*, and we use $\alpha, \beta, \gamma, \ldots$ to range over all steps $\mathcal{P}(T)$. Sometimes we identify a step α with its characteristic function $\alpha : T \to \{0, 1\}$, and then write $\alpha = \sum_{a \in T} \alpha(a) \cdot a$. The size $|\alpha|$ of α is defined as the number of its elements.

Definition 2.1. (step transition system)
A *step transition system* (or ST-system) over a set of net transitions T is a triple $\mathcal{S} = (Q, A, q_0)$ consisting of a set of *states* Q, including the initial state $q_0 \in Q$, and a set of labelled arcs $A \subseteq Q \times \mathcal{P}(T) \times Q$. It is assumed that:

- $(q, \varnothing, q) \in A$ for all $q \in Q$;
- the transition relation is deterministic, i.e., if $(q, \alpha, q') \in A$ and $(q, \alpha, q'') \in A$ then $q' = q''$;
- each state is reachable, i.e., if $q \in Q$ then there are sequences of steps $\alpha_1, \ldots, \alpha_n$ ($n \geq 0$) and states $q_1, \ldots, q_n = q$ such that $(q_{i-1}, \alpha_i, q_i) \in A$ for $1 \leq i \leq n$. ■

For an ST-system $\mathcal{S}$ as above, we introduce the following notations:

- $q \xrightarrow{\alpha} q'$ and $q \xrightarrow{\alpha}$ whenever $(q, \alpha, q') \in A$;
- $En_{\mathcal{S}}(q) = \{\alpha \mid q \xrightarrow{\alpha}\} \subseteq \mathcal{P}(T)$ is the set of all steps enabled at a state q;

- $En_{\mathcal{S}} = \bigcup_{q \in Q} En_{\mathcal{S}}(q) \subseteq \mathcal{P}(T)$ is the set of all the enabled steps of $\mathcal{S}$;
- $ready_{\mathcal{S}}(q) = \bigcup En_{\mathcal{S}}(q) = \bigcup_{\alpha \in En_{\mathcal{S}}(q)} \alpha \subseteq T$ is the set of all transitions ready to be executed at a state q, i.e., those belonging to steps enabled at q.

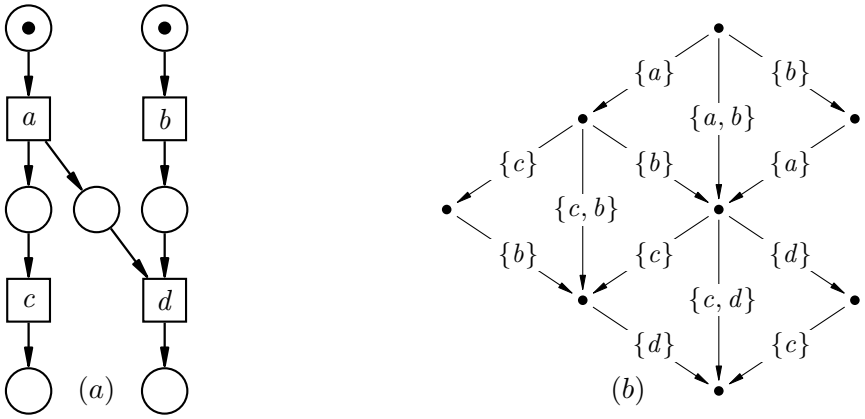


Figure 3: A PT-net $\mathcal{N}$ (a); and its concurrent reachability graph $CRG(\mathcal{N})$ (b). Note that arcs (in fact, self-loops) labelled by the empty step are not shown in $CRG(\mathcal{N})$.

Definition 2.2. (place/transition net)
A *place/transition net* (or PT-net) is a tuple $\mathcal{N} = (P, T, W, M_0)$, where P and T are finite disjoint sets of respectively *places* and *transitions*, $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is an arc weight function, and $M_0 : P \to \mathbb{N}$ is an *initial marking* (in general, any mapping $M : P \to \mathbb{N}$ is a marking). ■

We use the standard conventions concerning the graphical representation of PT-nets, as shown in Figure 3(a). For every element $x \in P \cup T$, we use

$$^\bullet x = \{y \mid W(y,x) > 0\} \quad \text{and} \quad x^\bullet = \{y \mid W(x,y) > 0\}$$

to denote the pre-set and post-set of x, respectively. If $x \in T$, then any $p \in {}^\bullet x$ is a pre-place of x, and any $p \in x^\bullet$ is a post-place. The dot-notation extends in the usual way to a set $X \subseteq P \cup T$:

$$^\bullet X = \bigcup_{x \in X} {}^\bullet x \quad \text{and} \quad X^\bullet = \bigcup_{x \in X} x^\bullet \, .$$

Moreover, for every place $p \in P$ and step $\alpha \in \mathcal{P}(T)$, we denote:

$$W(p,\alpha) = \sum_{a \in T} \alpha(a) \cdot W(p,a) \quad \text{and} \quad W(\alpha,p) = \sum_{a \in T} \alpha(a) \cdot W(a,p) \, .$$

In other words, $W(p,\alpha)$ gives the number of tokens that the execution of α removes from p, and $W(\alpha,p)$ is the total number of tokens inserted into p after the execution of α.

Definition 2.3. (place/transition net behaviour)
Let M be a marking of $\mathcal{N}$. A step $\alpha \in \mathcal{P}(T)$ is *enabled* and may be *executed* at M if, for every $p \in P$:

$$M(p) \geq W(p, \alpha) \ . \tag{1}$$

We denote this by $M[\alpha\rangle$. Executing such a step leads to the marking M', for every $p \in P$ defined by:

$$M'(p) = M(p) - W(p, \alpha) + W(\alpha, p) \ . \tag{2}$$

We denote this by $M[\alpha\rangle M'$. Moreover, for a singleton step $\alpha = \{a\}$, we write $M[a\rangle$ and $M[a\rangle M'$ rather than $M[\{a\}\rangle$ and $M[\{a\}\rangle M'$. ■

The *concurrent reachability graph* of $\mathcal{N}$ is the ST-system

$$CRG(\mathcal{N}) = ([M_0\rangle, A, M_0)$$

over T where:

$$[M_0\rangle = \{M_n \mid \exists \alpha_1, \ldots, \alpha_n, M_1, \ldots, M_{n-1} \ \forall 1 \leq i \leq n \ : \ M_{i-1}[\alpha_i\rangle M_i\} \tag{3}$$

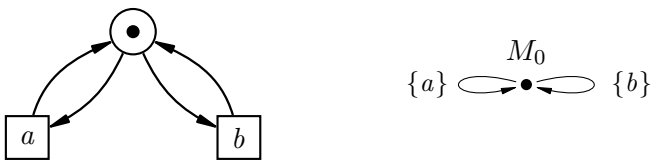
is the set of reachable markings and $(M, \alpha, M') \in A$ *iff* $M[\alpha\rangle M'$. Furthermore, we call $\alpha_1 \ldots \alpha_n$, as in the formula (3), a *step sequence* and write $M_0[\alpha_1 \ldots \alpha_n\rangle M_n$. Figure 3($b$) shows the concurrent reachability graph of the PT-net in Figure 3(a).

Remark 2.1. Note that for any Petri net $\mathcal{N}$, $CRG(\mathcal{N})$ is an ST-system, but we have different requirements for steps enabled in $CRG(\mathcal{N})$ and in an arbitrary ST-system. This is intentional distinction. In Section 5, we define sub-ST-systems, and a sub-ST-system of a CRG might not be a CRG of a Petri net.

Definition 2.4. (sequential and concurrent conflict)
Two distinct transitions, a and b, of $\mathcal{N}$ are in:

- *sequential conflict* at a marking M whenever $M[a\rangle$ and $M[b\rangle$, but $M[ab\rangle$ does not hold;
- *concurrent conflict* at a marking M whenever $M[a\rangle$ and $M[b\rangle$, but $M[\{a, b\}\rangle$ does not hold. ■

Note that sequential conflict implies concurrent conflict, but reverse implication does not hold; e.g., transitions a and b in Figure 4 are in concurrent conflict but not in a sequential one.



(a) (b)

Figure 4: A safe persistent PT-net $\mathcal{N}$ (a); and its concurrent reachability graph $CRG(\mathcal{N})$ (b).

Definition 2.5. (safe net)
$\mathcal{N}$ is *safe* if, for all reachable markings $M \in [M_0\rangle$ and places $p \in P$, $M(p) \leq 1$. ■

Note that reachable markings of safe nets can be treated as sets of places. Moreover, being a safe PT-net does not depend on the chosen semantics, i.e., the sequential semantics where only singleton steps are executed, or the full step semantics.

A step α of a PT-net $\mathcal{N}$ is:

- *active* if there is a reachable marking of $\mathcal{N}$ enabling it;
- *positive* if $W(\alpha, p) \geq W(p, \alpha)$, for every $p \in P$;
- *disconnected* if $({}^\bullet a \cup a^\bullet) \cap ({}^\bullet b \cup b^\bullet) = \varnothing$, for all distinct $a, b \in \alpha$;[1]
- *lying on self-loops* if $W(p, a) = W(a, p)$, for all $a \in \alpha$ and $p \in P$.

Clearly, the empty step is lying on self-loops, and if α is lying on self-loops then it is also positive.

Fact 2.1. (marking monotonicity)
If $M[\alpha\rangle$ and $M'(p) \geq M(p)$, for all $p \in P$, then $M'[\alpha\rangle$. ■

Fact 2.2. (step monotonicity)
If $M[\alpha\rangle$ and $\beta \subseteq \alpha$, then $M[\beta(\alpha \setminus \beta)\rangle$. ■

Fact 2.3. (disconnectedness)
A step α is enabled at a reachable marking M of a safe PT-net iff α is disconnected and consists of transitions enabled at M. ■

Remark 2.2. Three above facts are very intuitive and look trivial. However, they have a crucial importance in the discussion on persistence and nonviolence.

3. Step persistence in nets

Muller's speed independence theory provided a unique method for guaranteeing hazard-freeness of asynchronous circuits [17]. The *semimodularity* condition in this work required that an excitation of a circuit element must not be withdrawn before being absorbed by the system [21]. This condition was identified by Keller in [13] to be the same as the property of persistence[2] in a transition system model for asynchronous parallel computation. Thus satisfying the property of persistence became one of the key requirements when designing hazard-free asynchronous circuits.

Later, the idea of persistence was investigated in many papers, for example, in [1, 2, 3, 4, 7, 8, 16, 22]. However, with the exception of [7, 8], it was only considered in the context of sequential executions of systems, and defined for single transitions (rather than steps).

In the rest of this section, $\mathcal{N} = (P, T, W, M_0)$ is an arbitrary PT-net.

[1] The notion of disconnectedness as defined here is not related to the standard graph-theoretic notion of disconnectedness (i.e., that a graph is disconnected if there is a pair of vertices without a connecting path).
[2] [13] was the first work to consider persistence in the context of Petri nets.

Definition 3.1. (persistent net, [16])
$\mathcal{N}$ is *persistent* if, for all distinct transitions $a, b \in T$ and any reachable marking $M \in [M_0\rangle$, $M[a\rangle$ and $M[b\rangle$ imply $M[ab\rangle$. ■

We can re-write the net-oriented definition of persistence from the position of a single transition. Interestingly, such an attempt leads to two possible formalisations. In the first one, executing a *nonviolent* transition does not disable any other enabled transition, and in the second one, a *persistent* transition cannot be disabled by executing any other transition.

Definition 3.2. (nonviolent/persistent transition)
Let a be a transition enabled at a marking M of $\mathcal{N}$. Then a is *locally*:

- *nonviolent at* M if, for every transition b enabled at M, $b \neq a \implies M[ab\rangle$.
- *persistent at* M if, for every transition b enabled at M, $b \neq a \implies M[ba\rangle$.

Moreover, an active transition a is *globally nonviolent* (or *globally persistent*) in $\mathcal{N}$ if it is locally nonviolent (resp. locally persistent) at every reachable marking of $\mathcal{N}$ at which it is enabled. ■

The above net-oriented and transition-oriented definitions are closely related as, due to the symmetric roles played by a and b in Definition 3.2, we immediately obtain the following.

Proposition 3.1. The following are equivalent:

- $\mathcal{N}$ is persistent;
- $\mathcal{N}$ contains only globally nonviolent transitions;
- $\mathcal{N}$ contains only globally persistent transitions.

We now introduce the central definitions of this paper, in which we lift the notions of persistence and nonviolence from the level of individual transitions to the level of steps.

Definition 3.3. (nonviolent step in a net)
Let α be a step enabled at a marking M of $\mathcal{N}$. Then:

- α is *locally* A*-nonviolent at marking* M (or LA-nonviolent) if, for every step β enabled at M,
$$\beta \neq \alpha \implies M[\alpha(\beta \setminus \alpha)\rangle$$
- α is *locally* B*-nonviolent at marking* M (or LB-nonviolent) if, for every step β enabled at M,
$$\beta \cap \alpha = \varnothing \implies M[\alpha\beta\rangle$$
- α is *locally* C*-nonviolent at marking* M (or LC-nonviolent) if, for every step β enabled at M,
$$\beta \neq \alpha \implies M[\alpha\beta\rangle$$

Moreover, an active step α is *globally* A/B/C*-nonviolent* (or GA/GB/GC-nonviolent) in $\mathcal{N}$ if it is respectively LA/LB/LC-nonviolent at every reachable marking of $\mathcal{N}$ at which it is enabled. ■

Each of the three types of step nonviolence is a conservative extension of the transition nonviolence introduced in Definition 3.2. Intuitively, type-A nonviolence requires that only the unexecuted part of a delayed step is kept enabled, and so it is 'protected' by α. Type-B and type-C nonviolence, however, insist on maintaining the enabledness of whole delayed steps.

Definition 3.4. (persistent step in a net)
Let α be a step enabled at a marking M of $\mathcal{N}$. Then:

- α is *locally* A*-persistent at marking* M (or LA-persistent) if, for every step β enabled at M,
$$\beta \neq \alpha \implies M[\beta(\alpha \setminus \beta)\rangle$$
- α is *locally* B*-persistent at marking* M (or LB-persistent) if, for every step β enabled at M,
$$\beta \cap \alpha = \varnothing \implies M[\beta\alpha\rangle$$
- α is *locally* C*-persistent at marking* M (or LC-persistent) if, for every step β enabled at M,
$$\beta \neq \alpha \implies M[\beta\alpha\rangle$$

Moreover, an active step α is *globally* A/B/C*-persistent* (or GA/GB/GC-persistent) in $\mathcal{N}$ if it is respectively LA/LB/LC-persistent at every reachable marking of $\mathcal{N}$ at which it is enabled.

Again, each of the three types of step persistence is a conservative extension of transition persistence of Definition 3.2. Type-A persistence requires that only the unexecuted part of a delayed step is kept enabled, and in this case a persistent step can fail to fully 'survive'. Type-B and type-C persistence, however, insist on preserving the enabledness of whole steps. In type-B of nonviolence and persistence, two steps are considered distinct if they are disjoint, whereas in the other two cases it is enough that they are different, and so they can have a nonempty intersection. The empty step is trivially nonviolent and persistent according to all the nonviolence and persistence types in Definitions 3.3 and 3.4. Note also that although one could drop $\beta \neq \alpha$ in the definitions of type-A nonviolence and persistence, we decided to keep it in order to emphasize a link with the original notion of persistence introduced in [16].

> Since, as proven later, type-A and type-B nonviolence (as well as persistence) are equivalent, in the examples we discuss only the type-A and type-C variants of nonviolence and persistence.

Moving from sequential to step semantics changes the perception of persistence in PT-nets introduced by the standard Definition 3.1. In particular, in the sequential semantics, by Proposition 3.1, all transitions in a persistent net are both globally nonviolent and globally persistent. In the step semantics the situation is different. Consider, for example, the PT-net in Figure 5. It is persistent, and all of its active steps are both GA-persistent and GA-nonviolent. However, its nonempty steps fail to be LC-persistent or LC-nonviolent at some of the markings that enable them. More precisely, $\{a\}$, $\{b\}$ and $\{a, b\}$ are neither LC-persistent nor LC-nonviolent at M_0, while $\{c\}$, $\{d\}$ and $\{c, d\}$ are neither LC-persistent nor LC-nonviolent at $M_1 = \{p_5, p_6, p_7, p_8\}$. This should not come as a surprise, as type-C persistence and nonviolence are demanding properties. Type-A persistence and nonviolence, on the other hand, are close in spirit to their sequential counterparts.

A duality of nonviolence and persistence is illustrated in Figure 6, where:

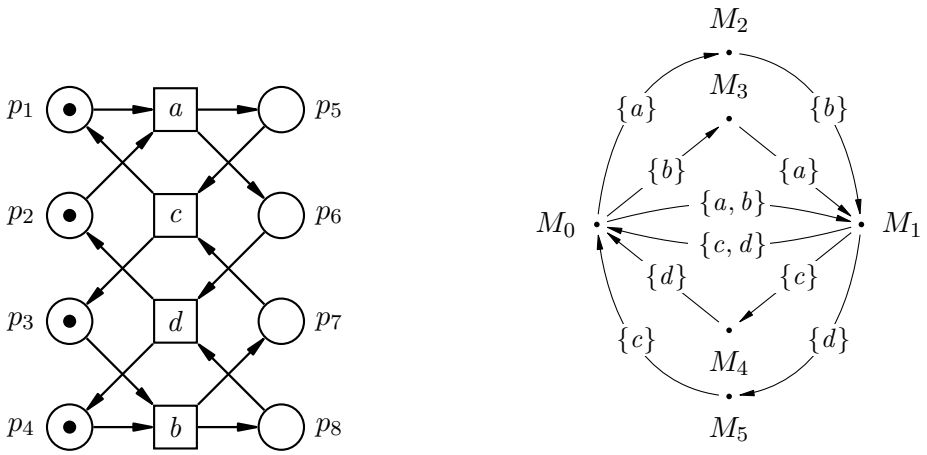


Figure 5: A safe persistent PT-net and its concurrent reachability graph.

- $\{a\}$ is both GA-nonviolent and GC-nonviolent, but neither LA-persistent nor LC-persistent at M_0,
- $\{b\}$ is both GA-persistent and GC-persistent, but neither LA-nonviolent nor LC-nonviolent at M_0.

A step can be both nonviolent and persistent. For example, if we merge the two places in Figure 6, then $\{a\}$ and $\{b\}$ become GA/GC-nonviolent/persistent.

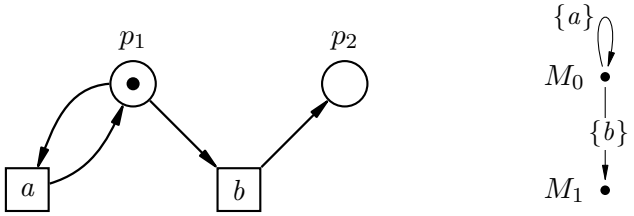


Figure 6: A safe PT-net illustrating the duality of persistence and nonviolence.

4. Relating persistent and nonviolent steps

In this section, we investigate the expressiveness of different notions of persistence and nonviolence defined for steps, assuming first that $\mathcal{N} = (P, T, W, M_0)$ is a general PT-net.

Proposition 4.1. Let α be a step enabled at a reachable marking M of $\mathcal{N}$.

1. If α is GA/GB/GC-nonviolent in $\mathcal{N}$, then α is LA/LB/LC-nonviolent at M.
2. If α is GA/GB/GC-persistent in $\mathcal{N}$, then α is LA/LB/LC-persistent at M.

Proof:
Follows directly from Definitions 3.3 and 3.4. □

We then obtain a number of inclusions between different types of persistent and nonviolent steps.

Proposition 4.2. Let α be an active step and M be a marking of $\mathcal{N}$. Then:

1. α is LA-nonviolent at M iff α is LB-nonviolent at M.
2. α is LA-persistent at M iff α is LB-persistent at M.

Proof:
Assume that α is enabled at M, and β is another step enabled at M.

(1) Suppose that α is LA-nonviolent at M and $\beta \cap \alpha = \varnothing$. Then $M[\alpha(\beta \setminus \alpha)\rangle$ and $\beta \setminus \alpha = \beta$. Hence $M[\alpha\beta\rangle$, and so α is LB-nonviolent at M.
Conversely, suppose α is LB-nonviolent at M and $\beta \neq \alpha$. Then $M[\alpha(\beta \setminus \alpha)\rangle$ as $(\beta \setminus \alpha) \cap \alpha = \varnothing$ and $M[\beta \setminus \alpha\rangle$ (cf. Fact 2.2). Hence, α is LA-nonviolent at M.

(2) Suppose that α is LA-persistent at M and $\beta \cap \alpha = \varnothing$. Then $M[\beta(\alpha \setminus \beta)\rangle$ and $\alpha \setminus \beta = \alpha$. Hence $M[\beta\alpha\rangle$, and so α is LB-persistent at M.
Conversely, suppose that α is LB-persistent at M and $\beta \neq \alpha$. Then $M[(\beta \setminus \alpha)\alpha\rangle$ as $(\beta \setminus \alpha) \cap \alpha = \varnothing$ and $M[\beta \setminus \alpha\rangle$ (cf. Fact 2.2). Hence $M[(\beta \setminus \alpha)(\alpha \cap \beta)(\alpha \setminus \beta)\rangle$ (cf. Fact 2.2). Thus, by $M[\beta\rangle$, $M[\beta(\alpha \setminus \beta)\rangle$. Hence α is LA-persistent at M. □

Corollary 4.1. Let α be an active step of $\mathcal{N}$. Then:

1. α is GA-nonviolent in $\mathcal{N}$ iff α is GB-nonviolent in $\mathcal{N}$.
2. α is GA-persistent in $\mathcal{N}$ iff α is GB-persistent in $\mathcal{N}$.

Proposition 4.3. Let α be an active step and M a marking of $\mathcal{N}$. Then:

1. If α is LC-nonviolent at M, then α is LA-nonviolent at M.
2. If α is LC-persistent at M, then α is LA-persistent at M.

Proof:
Since enabledness of steps is monotonic in PT-nets (cf. Fact 2.2), the two implications follow directly from Definitions 3.3 and 3.4, where the statements for LC-persistence and LC-nonviolence have stronger consequents. □

Corollary 4.2. Let α be an active step of a $\mathcal{N}$. Then:

1. If α is GC-nonviolent in $\mathcal{N}$, then α is GA-nonviolent in $\mathcal{N}$.
2. If α is GC-persistent in $\mathcal{N}$, then α is GA-persistent in $\mathcal{N}$.

We now make a series of statements and observations concerning the general PT-net model.

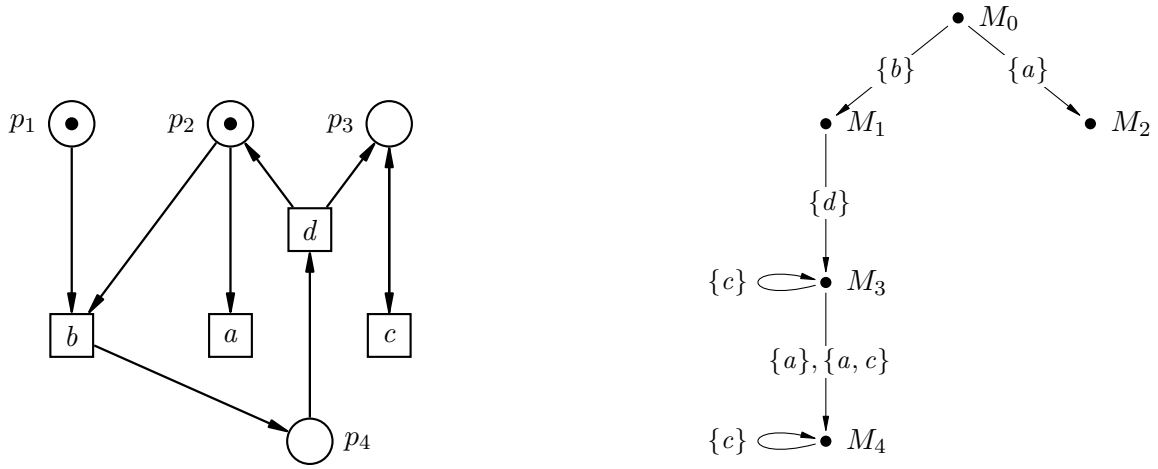


Figure 7: A safe PT-net and its concurrent reachability graph.

- The implications in Proposition 4.3 cannot be reversed. A counterexample is provided in Figure 7, where $\{a\}$ is both LA-nonviolent and LA-persistent at $M_3 = \{p_2, p_3\}$. However, it is neither LC-nonviolent nor LC-persistent at M_3. The example in Figure 7 can as well be an illustration for Proposition 4.1 (type-A) showing the case of an LA-nonviolent and an LA-persistent step $\{a\}$ (at M_3), which is neither GA-nonviolent nor GA-persistent (because of M_0).
- The implications in Corollary 4.2 cannot be reversed. A counterexample is again provided in Figure 7, where $\{a, c\}$ is both GA-nonviolent and GA-persistent, but it is neither GC-nonviolent nor GC-persistent. As this step is only enabled at marking M_3, it fails to be LC-nonviolent or LC-persistent as well. Moreover, in Figure 7, $\{d\}$ is a step that is type-A and type-C globally nonviolent and persistent, because it is only enabled at one marking, M_1, and no other nonempty step is enabled at M_1.
- The top PT-net in Figure 8 shows that a step $\{a\}$ may be GA-persistent, but only LC-persistent (at M_4). Step $\{a\}$ is not GC-persistent, because it is not LC-persistent at M_2. The same example can be used when considering nonviolence.
- The middle PT-net in Figure 8 shows an example of a step, $\{a\}$, that is LC-nonviolent and LC-persistent at M_0 (hence also LA-nonviolent and LA-persistent at M_0), but it is neither GA-nonviolent nor GA-persistent (consequently neither GC-nonviolent nor GC-persistent).
- There may be steps in PT-nets that fail to satisfy all the types of persistence and nonviolence; for example, $\{a, c\}$ and $\{b\}$ in the bottom PT-net of Figure 8.
- There are PT-nets where all steps are neither persistent nor nonviolent whatever type (A or C) we choose. For example, take the bottom PT-net in Figure 8 and delete p_2, p_5 and c with all adjacent arcs. Then, the only nonempty steps in the concurrent reachability graph are $\{a\}$ and $\{b\}$, and they prevent each other from being persistent. As a result, they also fail to be nonviolent.

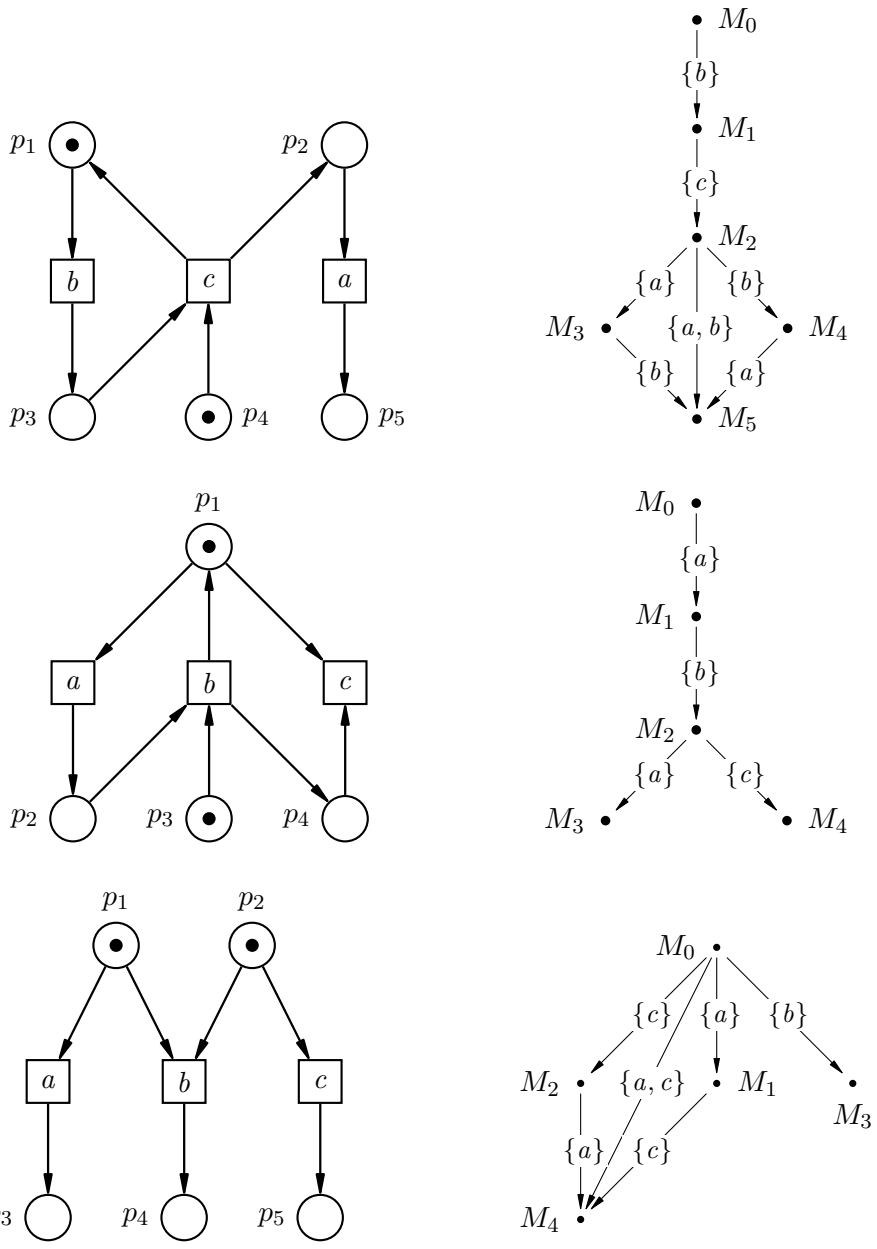


Figure 8: Three safe PT-nets and their concurrent reachability graphs.

4.1. Global persistence and nonviolence in safe PT-nets

In this section, we focus our attention on safe PT-nets, and assume throughout that $\mathcal{N} = (P, T, W, M_0)$ is such a net.

It turns out that all non-singleton steps in $\mathcal{N}$, which are GC-persistent or GC-nonviolent, are built out

of transitions lying on self-loops. To show this, we first prove an auxiliary result.

Proposition 4.4. Let α be a GC-persistent or GC-nonviolent step enabled at a reachable marking M of a safe PT-net $\mathcal{N}$. Then ${}^\bullet(\alpha \cap \beta) = (\alpha \cap \beta)^\bullet$, for every step $\beta \neq \alpha$ enabled at M.

Proof:
We may assume that $\alpha \cap \beta \neq \varnothing$ as for $\alpha \cap \beta = \varnothing$ the result holds by ${}^\bullet(\alpha \cap \beta) = \varnothing = (\alpha \cap \beta)^\bullet$.

Assume that α is GC-persistent (or respectively GC-nonviolent) and consider two cases.

Note that by Fact 2.3 steps α, β and $\alpha \cap \beta$ are disconnected.

Case 1: $p \in {}^\bullet(\alpha \cap \beta)$, for some step $\beta \neq \alpha$ enabled at M. Clearly, $M(p) = 1$ and there exists $b \in (\alpha \cap \beta)$ such that $p \in {}^\bullet b$.

If α is GC-persistent, there is a marking M' such that $M[\beta\rangle M'[\alpha\rangle$. As $M'[\alpha\rangle$ and $p \in {}^\bullet(\alpha \cap \beta)$, we have $M'(p) = 1$. Furthermore, $M'(p) = M(p) - W(p, \beta) + W(\beta, p)$. On the other hand, if α is GC-nonviolent, there is a marking M' such that $M[\alpha\rangle M'[\beta\rangle$. As $M'[\beta\rangle$ and $p \in {}^\bullet(\alpha \cap \beta)$, we have $M'(p) = 1$. Furthermore, $M'(p) = M(p) - W(p, \alpha) + W(\alpha, p)$.

And so, in both cases, by disconnectedness of α and β, and the fact that $M(p) = M'(p) = 1$ we get $W(b, p) = W(p, b) = 1$. Moreover, for any $c \in (\alpha \cup \beta)$ such that $c \neq b$ we get $W(c, p) = W(p, c) = 0$. Therefore $W(p, a) = W(a, p)$ for each $a \in (\alpha \cup \beta)$. Hence $p \in (\alpha \cap \beta)^\bullet$, and so ${}^\bullet(\alpha \cap \beta) \subseteq (\alpha \cap \beta)^\bullet$.

Case 2: $p \in (\alpha \cap \beta)^\bullet \setminus {}^\bullet(\alpha \cap \beta)$. Then, by $M[\alpha \cap \beta\rangle$ and the safeness of $\mathcal{N}$, $M(p) = 0$. Hence, by $M[\alpha\rangle$ and $M[\beta\rangle$, we must have $p \notin {}^\bullet\alpha \cup {}^\bullet\beta$. Consequently, since there is M'' such that $M[\beta\alpha\rangle M''$ in case of GC-persistence of α and $M[\alpha\beta\rangle M''$ in case of GC-nonviolence of α, we obtain $M''(p) \geq 2$, a contradiction with $\mathcal{N}$ being safe. Thus $(\alpha \cap \beta)^\bullet \setminus {}^\bullet(\alpha \cap \beta) = \varnothing$.

Hence ${}^\bullet(\alpha \cap \beta) = (\alpha \cap \beta)^\bullet$ and the result holds. □

Theorem 4.1. Let α be a non-singleton active step of a safe PT-net $\mathcal{N}$. If α is GC-persistent or GC-nonviolent, then it is lying on self-loops.

Proof:
If $\alpha = \varnothing$ the result holds. Let $|\alpha| \geq 2$. Suppose that $a \in \alpha$ and M be a reachable marking enabling α. Since $\{a\} \neq \alpha$ and $M[a\rangle$ for any marking M such that $M[\alpha\rangle$, we have ${}^\bullet(\alpha \cap \{a\}) = (\alpha \cap \{a\})^\bullet$ (cf. Proposition 4.4). Hence ${}^\bullet a = a^\bullet$. □

We now want to relate the persistence and nonviolence of a step with the persistence (resp. nonviolence) of its constituent transitions in safe nets. We first consider GA-persistent and GA-nonviolent steps, but as we already know, from Corollary 4.1, the results would also hold for GB-persistent (resp. GB-nonviolent) steps.

Theorem 4.2. Let α be an active step in a safe PT-net $\mathcal{N}$. If all the transitions in α are globally persistent (or globally nonviolent), then α is GA-persistent (resp. GA-nonviolent).

Proof:
Let M be a reachable marking and $\beta \neq \alpha$ be a step in $\mathcal{N}$ such that $M[\alpha\rangle$ and $M[\beta\rangle$.

We first assume that all the transitions in α are globally persistent. We need to show that $M[\beta(\alpha \setminus \beta)\rangle$.

Let $\alpha \cap \beta = \{a_1, \ldots, a_m\}$, $\alpha \setminus \beta = \{b_1, \ldots, b_n\}$ and $\beta \setminus \alpha = \{c_1, \ldots, c_k\}$. Note that all the transitions in these three sets are different. From $M[\beta\rangle$ and Fact 2.2, we have $M[a_1 \ldots a_m c_1 \ldots c_k\rangle$. Now, since each b_i is globally persistent and enabled at M, we have that $M[a_1 \ldots a_m c_1 \ldots c_k b_1 \ldots b_n\rangle$. Since α and β are steps in a safe net $\mathcal{N}$ enabled at some marking (M), we have, from Fact 2.3, that transitions in α and β have disjoint pre-sets and post-sets. Hence we have $M[\beta(\alpha \setminus \beta)\rangle$.

Assume now that all the transitions in α are globally nonviolent. Note that, by Fact 2.3, β is disconnected. We need to show that $M[\alpha(\beta \setminus \alpha)\rangle$.

Let $\alpha = \{a_1, \ldots, a_m\}$ and $\beta \setminus \alpha = \{c_1, \ldots, c_k\}$. From $M[\alpha\rangle$ and Fact 2.2, we have $M[a_1 \ldots a_m\rangle M'$. Now, for each transition c_i, since $M[c_i\rangle$ and every a_i is globally nonviolent, we have that $M'[c_i\rangle$. Thus, by Fact 2.3, $M'[\beta \setminus \alpha\rangle$, and so $M[\alpha(\beta \setminus \alpha)\rangle$. □

We now consider GC-persistent steps. In this case the antecedent in the implication is stronger.

Theorem 4.3. Let α be an active step in a safe PT-net $\mathcal{N}$. If all the transitions in α are globally persistent and lying on self-loops, then α is GC-persistent.

Proof:
Let M be a reachable marking and $\beta \neq \alpha$ be a step such that $M[\alpha\rangle$ and $M[\beta\rangle$. We need to show that $M[\beta\alpha\rangle$.
Let $\alpha \cap \beta = \{a_1, \ldots, a_m\}$, $\alpha \setminus \beta = \{b_1, \ldots, b_n\}$ and $\beta \setminus \alpha = \{c_1, \ldots, c_k\}$. Proceeding similarly as in the previous proof we can show that

$$M[a_1 \ldots a_m c_1 \ldots c_k b_1 \ldots b_n\rangle \ .$$

Since all transitions in α are lying on self-loops and are globally persistent, we further obtain

$$M[a_1 \ldots a_m c_1 \ldots c_k a_1 \ldots a_m b_1 \ldots b_n\rangle \ .$$

Hence, from $M[\alpha\rangle$, $M[\beta\rangle$ and Fact 2.3, we have $M[\beta\alpha\rangle$. □

In Theorems 4.2 and 4.3, the reverse implications do not hold. A counterexample is provided in Figure 9, where a persistent step $\alpha = \{a, c\}$ contains a non-globally persistent transition a. Indeed, $\alpha = \{a, c\}$ is both GA-persistent and GC-persistent, but $a \in \alpha$ is not persistent at $M_0 = \{p_1, p_2\}$, because there exists $b \neq a$ such that $M_0[a\rangle$ and $M_0[b\rangle$, but $M_0[ba\rangle$ does not hold. A counterexample for 'non-reversibility' of Theorem 4.2 in the case of GA-nonviolence can be found in Figure 7, where a GA-nonviolent step $\{a, c\}$ contains transition a that is not globally nonviolent (because of M_0).

The situation is slightly different in the case of GC-nonviolence.

Theorem 4.4. Let α be an active step of a safe PT-net $\mathcal{N}$. If all the transitions in α are lying on self-loops, then it is GC-nonviolent.

Proof:
Since all the transitions in α are lying on self-loops, for every marking M such that $M[\alpha\rangle$, we have $M[\alpha\rangle M$. Hence α is GC-nonviolent in $\mathcal{N}$. □

Corollary 4.3. Let α be a non-singleton active step of a safe PT-net $\mathcal{N}$. Then α is GC-nonviolent if and only if α is lying on self-loops.

Proof:
Follows from Theorems 4.4 and 4.1. □

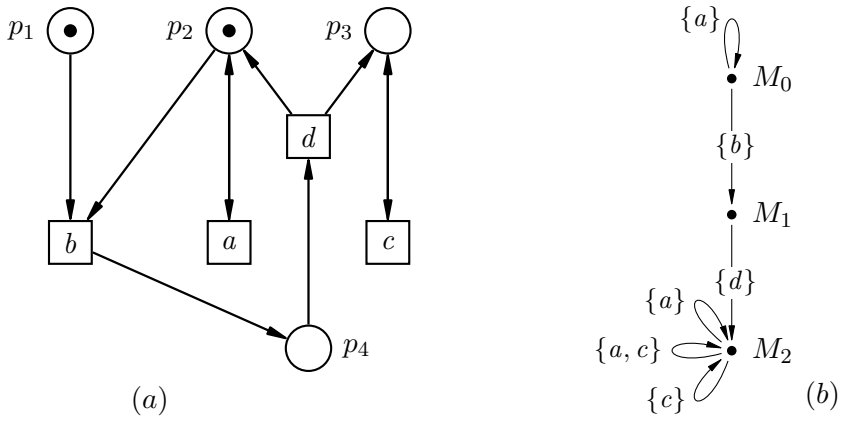


Figure 9: A safe PT-net $\mathcal{N}$ (a); and its concurrent reachability graph $CRG(\mathcal{N})$ (b).

5. Pruning reachability graphs

In this section, we turn from general considerations relating to the nonviolence and persistence of active steps to more application oriented discussion, restricting ourselves to the case of global persistence.

The original motivation for studying persistent steps in this paper was to discover which sets of transitions — called later *bundles* — can be executed synchronously and therefore be treated as some kind of 'atomic actions', giving rise to new 'bigger' transitions, which would execute in a 'hazard-free' way. In the application area of asynchronous circuits, bundling actions would reduce signal management by merging concurrent signals into one event. This merging must be done in a consistent fashion. The best candidates for bundles are persistent steps, but if we want to form 'bigger' transitions from them, we must make sure that one enabled persistent step does not include another enabled persistent step. All the transitions in a bundle must always appear together, in the same configurations. In the ideal situation (we say ideal, because it might be difficult to achieve), we do not want to allow, for example, three persistent steps $\{a, b\}$, $\{a\}$ and $\{b\}$ to be enabled in a given transition system. We need to choose: either to opt for $\{a, b\}$ and delete $\{a\}$ and $\{b\}$, or the other way round. Therefore, we need to develop an algorithm which, for a given net $\mathcal{N}$, would allow us to prune its reachability graph $CRG(\mathcal{N})$ in such a way that all persistent steps would satisfy an additional 'non-inclusion' condition. The 'pruned' transition system would represent the desired behaviour, which then we would like to implement in a form of a Petri net in a process of synthesis. We start by defining sub-ST-systems which will be obtained by pruning concurrent reachability graphs.

Definition 5.1. (sub-ST-system)
An ST-system $\mathcal{S} = (Q, A, q_0)$ is a *sub*-ST-*system* of an ST-system $\mathcal{S}' = (Q', A', q_0)$ if $Q \subseteq Q'$, $A \subseteq A'$ and, for every $q \in Q$, $ready_{\mathcal{S}}(q) = ready_{\mathcal{S}'}(q)$. We denote this by $\mathcal{S} \preccurlyeq \mathcal{S}'$. ■

In the above definition, $En_{\mathcal{S}}$ of a 'properly pruned' reachability graph $\mathcal{S}'$ will be a set of bundles. What we mean by 'properly pruned' will be described by conditions stated in Problem 1.

We now re-define for ST-systems the three notions concerned with global persistence introduced for PT-nets. The reason is that once we start pruning an ST-system, we need to check whether the remaining

steps that were previously persistent remain persistent. Such checks will be carried out for ST-systems that might not be concurrent reachability graphs of any PT-nets.

Definition 5.2. (persistent step in an ST-system)
A step $\alpha \in En_{\mathcal{S}}$ is GA/GB/GC-*persistent* in an ST-system $\mathcal{S} = (Q, A, q_0)$ if, for all states $q \in Q$ and steps β such that $\alpha, \beta \in En_{\mathcal{S}}(q)$ we respectively have:

$$\begin{array}{lrcl} (\text{GA}) & \beta \neq \alpha & \Longrightarrow & q \xrightarrow{\beta(\alpha\setminus\beta)} \\ (\text{GB}) & \beta \cap \alpha = \varnothing & \Longrightarrow & q \xrightarrow{\beta\alpha} \\ (\text{GC}) & \beta \neq \alpha & \Longrightarrow & q \xrightarrow{\beta\alpha} \ . \end{array}$$

■

Proposition 5.1. A step α is GA/GB/GC-persistent in a PT-net iff α is respectively GA/GB/GC-persistent in its concurrent reachability graph.

Proof:
Follows directly from the definitions. □

We have the following relationships between just introduced notions of step persistence.

Proposition 5.2. Let $\mathcal{S} = (Q, A, q_0)$ be an ST-system.

1. If $\alpha \in En_{\mathcal{S}}$ is GA-persistent, then it is also GB-persistent.
2. If $\alpha \in En_{\mathcal{S}}$ is GC-persistent, then it is also GB-persistent.

Proof:
(1) Let $q \in Q$ and $q \xrightarrow{\alpha}$ and $q \xrightarrow{\beta}$ be such that $\beta \cap \alpha = \varnothing$. Since $\alpha \in En_{\mathcal{S}}$ is GA-persistent, we have $q \xrightarrow{\beta(\alpha\setminus\beta)}$. Hence $q \xrightarrow{\beta\alpha}$ which means that $\alpha \in En_{\mathcal{S}}$ is GB-persistent.

(2) Follows directly from Definition 5.2. □

Unlike for PT-nets, in the case of ST-systems, GB-persistence does not imply GA-persistence of steps. Indeed, let $\alpha \in En_{\mathcal{S}}$ be a GB-persistent step in $\mathcal{S}$, and $\beta \neq \alpha$ and $q \in Q$ be such that $q \xrightarrow{\alpha}$ and $q \xrightarrow{\beta}$. We know that $\beta \cap (\alpha \setminus \beta) = \varnothing$. However, with such assumptions, we cannot in general guarantee that $q \xrightarrow{\alpha\setminus\beta}$. Though the latter is true for the concurrent reachability graphs of PT-nets, we must also consider ST-systems resulting from their pruning (see the ST-system depicted on Figure 11(c), where the step $\{b, d\}$ is GB-persistent, but not GA-persistent). For similar reasons, in the case of ST-systems, GC-persistence does not imply GA-persistence.

We now can formulate a problem which is our main concern in this section.

Problem 1. Let $\mathcal{N}$ be a PT-net and $CRG(\mathcal{N})$ be its concurrent reachability graph. Construct an ST-system $\mathcal{S}$ such that $\mathcal{S} \preccurlyeq CRG(\mathcal{N})$ and all steps in $En_{\mathcal{S}}$ are GB-persistent in $\mathcal{S}$[3], and additionally satisfying (GNI) or (LNI), where the latter conditions are defined as follows:

(GNI) $\alpha \not\subset \beta$, for all nonempty steps $\alpha, \beta \in En_{\mathcal{S}}$

(LNI) $\alpha \not\subset \beta$, for all states q and all nonempty steps $\alpha, \beta \in En_{\mathcal{S}}(q)$.

We denote this respectively by

$$\mathcal{S} \preccurlyeq_{pers}^{global} CRG(\mathcal{N}) \quad \text{and} \quad \mathcal{S} \preccurlyeq_{pers}^{local} CRG(\mathcal{N}) .$$

We also refer to (GNI) as *global non-inclusion*, and to (LNI) as *local non-inclusion*. ■

The difference between $\preccurlyeq_{pers}^{global}$ and $\preccurlyeq_{pers}^{local}$ is that the latter only requires non-inclusion of bundles locally for each state, whereas the former insists that non-inclusion holds globally.

Proposition 5.3. $\mathcal{S} \preccurlyeq_{pers}^{global} CRG(\mathcal{N})$ implies $\mathcal{S} \preccurlyeq_{pers}^{local} CRG(\mathcal{N})$.

Proof:
Follows directly from the definition. □

In our first attempt to solve Problem 1, we will concentrate on PT-nets that are persistent according to Definition 3.1. We then have the following result.

Theorem 5.1. Let $\mathcal{N}$ be a PT-net which is persistent according to Definition 3.1. Then there is an ST-system $\mathcal{S}$ satisfying $\mathcal{S} \preccurlyeq_{pers}^{global} CRG(\mathcal{N})$.

Proof:
It suffices to take $CRG(\mathcal{N})$ and delete all nonempty non-singleton steps. □

As the above proof produces completely sequential solution, we call such pruning to be trivial. We will now search for nontrivial, hence more concurrent ones. We will also require that the original PT-net is not only persistent, but also safe.

Proposition 5.4. Let $\mathcal{N}$ be a safe PT-net which is persistent according to Definition 3.1. Then all active steps in $\mathcal{N}$ are GB-persistent in $CRG(\mathcal{N})$.

Proof:
Let $\alpha \in En_{CRG(\mathcal{N})}$. As $\mathcal{N}$ is persistent according to Definition 3.1, all transitions in α are globally persistent according to Definition 3.2. Hence, from Theorem 4.2 and the fact that $\mathcal{N}$ is safe, we have that α is GA-persistent in $\mathcal{N}$, and also GB-persistent in $\mathcal{N}$ (see Corollary 4.1). Following Proposition 5.1, we conclude that α is GB-persistent in $CRG(\mathcal{N})$. □

[3] Alternatively, we could require GA-persistence or GC-persistence. We opted here for GB-persistence, because it is the weakest of the three notions.

The last result guarantees the GB-persistence of steps in the concurrent reachability graph of a safe persistent PT-net $\mathcal{N}$, but the non-inclusion conditions ((GNI) and (LNI)) are not, in general, satisfied in $CRG(\mathcal{N})$ due to Fact 2.2. To satisfy the non-inclusion conditions, we need to prune $CRG(\mathcal{N})$, but in such a way that GB-persistence of steps is maintained. We now explore what happens if we choose to prune all but the maximal steps at every reachable marking.

In what follows, the ST-system $CRG^{max}(\mathcal{N})$ is obtained from $CRG(\mathcal{N})$, the concurrent reachability graph of a PT-net $\mathcal{N}$, by deleting at every reachable marking M, all the arcs labelled by non-maximal nonempty steps (we do not delete the empty steps for technical reasons), and then removing those nodes that became unreachable from the initial state by the removal of such steps.

Proposition 5.5. $CRG^{max}(\mathcal{N}) \preccurlyeq CRG(\mathcal{N})$.

Proof:
Follows directly from the definitions and the fact that, for each enabled step, there is a maximal step enabled at the same marking. □

Proposition 5.6. $CRG^{max}(\mathcal{N})$ satisfies (LNI) in Problem 1.

Proof:
Follows from the fact that maximal nonempty steps are incomparable. □

Figures 10 and 11 show examples of persistent and safe PT-nets for which the described pruning procedure works as their $CRG^{max}(\mathcal{N})$ graphs contain only GB-persistent steps. In all these examples the pruned reachability graph satisfies $CRG^{max}(\mathcal{N}) \preccurlyeq^{local}_{pers} CRG(\mathcal{N})$, and in case of the example in Figure 10(a), we even have $CRG^{max}(\mathcal{N}) \preccurlyeq^{global}_{pers} CRG(\mathcal{N})$. So, the proposed pruning procedure helped to achieve local non-inclusion without jeopardising GB-persistence of the remaining steps. However, in Figures 10(f) and 11(c), the persistence in initial markings is achieved only because the steps enabled there are not disjoint, and so type-B persistence holds trivially.

In general, pruning non-maximal steps may make some of the remaining steps non-GB-persistent. Figure 12(c) shows that the initially enabled step $\{b\}$ is not GB-persistent after the pruning procedure, as after executing $\{a\}$ it is not longer enabled. Instead step $\{b, c\}$ is enabled, because it was the maximal step in the marking M. We therefore propose a weaker version of condition (GB) which holds for safe and persistent PT-nets.

Proposition 5.7. Let $\mathcal{N}$ be a safe PT-net which is persistent according to Definition 3.1. Then, for every marking M in $CRG^{max}(\mathcal{N})$, $M \xrightarrow{\alpha}$ and $M \xrightarrow{\beta}$ implies:

$$(\text{GB}') \quad \beta \cap \alpha = \varnothing \quad \Longrightarrow \quad \exists \gamma : \ \alpha \subseteq \gamma \wedge M \xrightarrow{\beta\gamma} \ .$$

Proof:
From Proposition 5.4 we know that $M \xrightarrow{\beta\alpha}$ in $CRG(\mathcal{N})$. Moreover, there is a maximal step γ available (as it is not removed by the pruning) after executing β from M such that $\alpha \subseteq \gamma$. Hence $M \xrightarrow{\beta\gamma}$ in $CRG^{max}(\mathcal{N})$. □

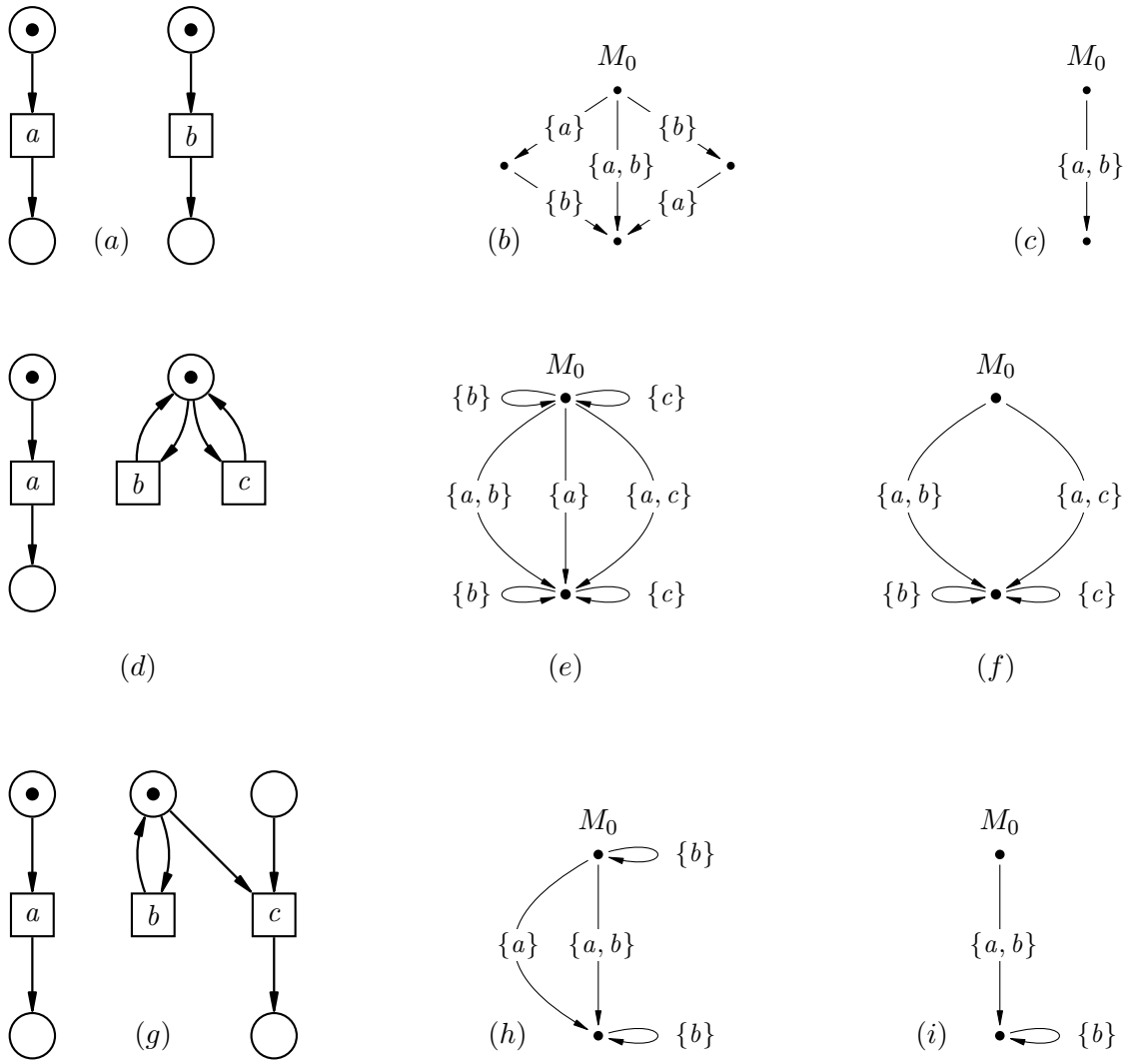


Figure 10: Three safe persistent PT-nets $\mathcal{N}$ (a, d, g); their concurrent reachability graphs $CRG(\mathcal{N})$ (b, e, h); and the corresponding $CRG^{max}(\mathcal{N}) \preccurlyeq^{local}_{pers} CRG(\mathcal{N})$ obtained in the pruning procedure (c, f, i).

Hence, pruning non-maximal steps may result in the loss of persistence when $\alpha \subset \gamma$ in (GB′). In such a case we may, however, 'repair' $\mathcal{N}$ by disabling γ. The mechanism for achieving this is simple, namely we select one transition from α, one transition from $\gamma \setminus \alpha$, and make sure that they cannot be executed in the same step.

Let $\mathcal{N}$ be a PT-net and $x \neq y$ be two transitions. Then $\mathcal{N}_{x \leftrightarrow y}$ is obtained from $\mathcal{N}$ by adding a new place p marked with one token, and such that $W(p, x) = W(x, p) = W(p, y) = W(y, p) = 1$. This construction is illustrated in Figure 12(d, e), where we try to fix the problem of the net $\mathcal{N}$ in Figure 12(a). We added a new place p and chose b and c to play the roles of x and y (the only choice in this example)

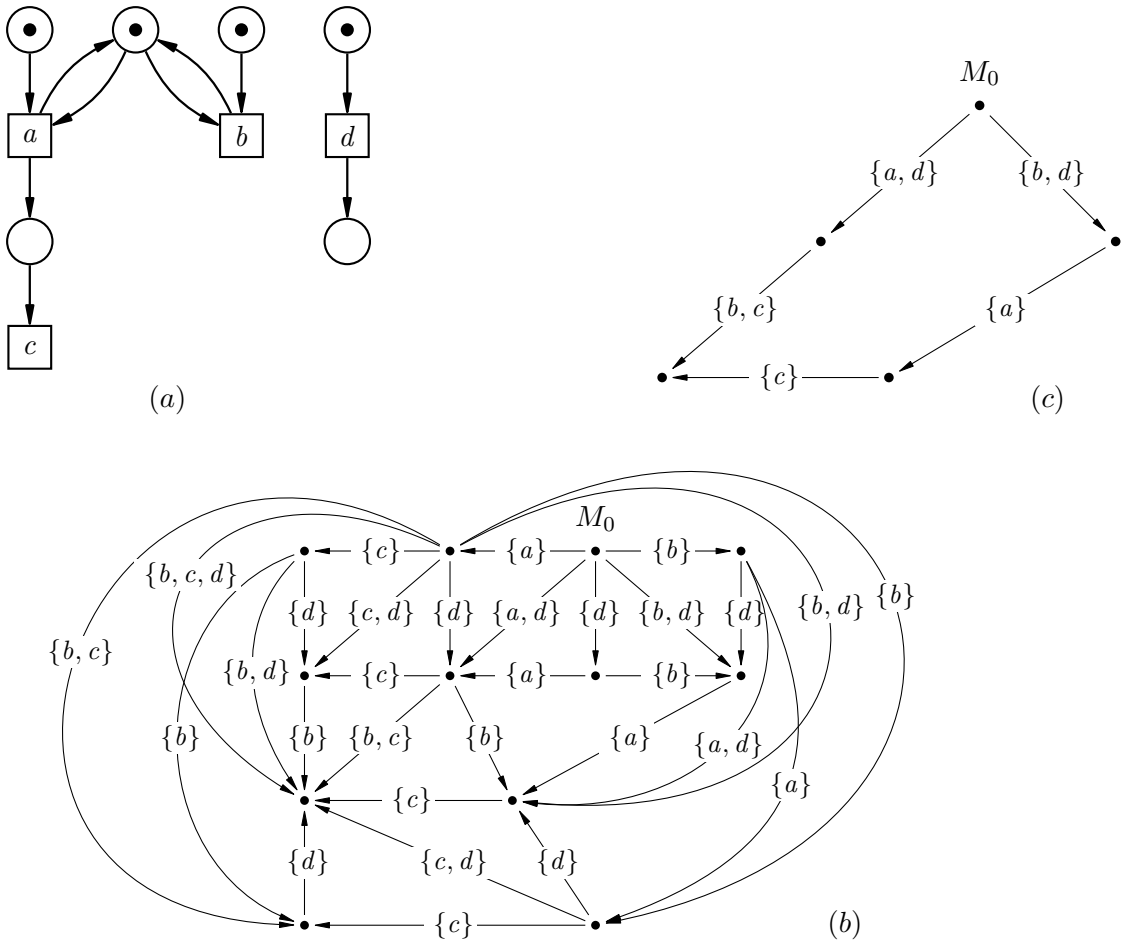


Figure 11: A safe persistent PT-net $\mathcal{N}$ (a); its concurrent reachability graph $CRG(\mathcal{N})$ (b); and $CRG^{max}(\mathcal{N}) \preccurlyeq^{local}_{pers} CRG(\mathcal{N})$ obtained in the pruning procedure (c).

creating $\mathcal{N}' = \mathcal{N}_{b \leftrightarrow c}$. The new place disables the concurrent step $\{b, c\}$ at M, leaving enabled only the singleton steps $\{b\}$ and $\{c\}$. They are now maximal steps at M. In fact, in this simple example, we have only singleton steps in the concurrent reachability graph of $\mathcal{N}'$, and so the pruning is trivial.

In the following propositions we show that after the proposed modification the PT-net generates a concurrent reachability graph which is a sub-ST-system of the reachability graph of the initial net. Also, the modified net is still safe and persistent according to Definition 3.1.

The two following facts result directly from definitions:

Fact 5.1. Let $\mathcal{N}$ be a safe PT-net which is persistent according to Definition 3.1. Then we have

$$CRG(\mathcal{N}_{x \leftrightarrow y}) \preccurlyeq CRG(\mathcal{N})$$

Moreover, the reachable markings of $CRG(\mathcal{N}_{x \leftrightarrow y})$ and $CRG(\mathcal{N})$ are the same, if we identify each reachable marking M of $\mathcal{N}$ with the marking $M \cup \{p\}$ of $\mathcal{N}_{x \leftrightarrow y}$. ■

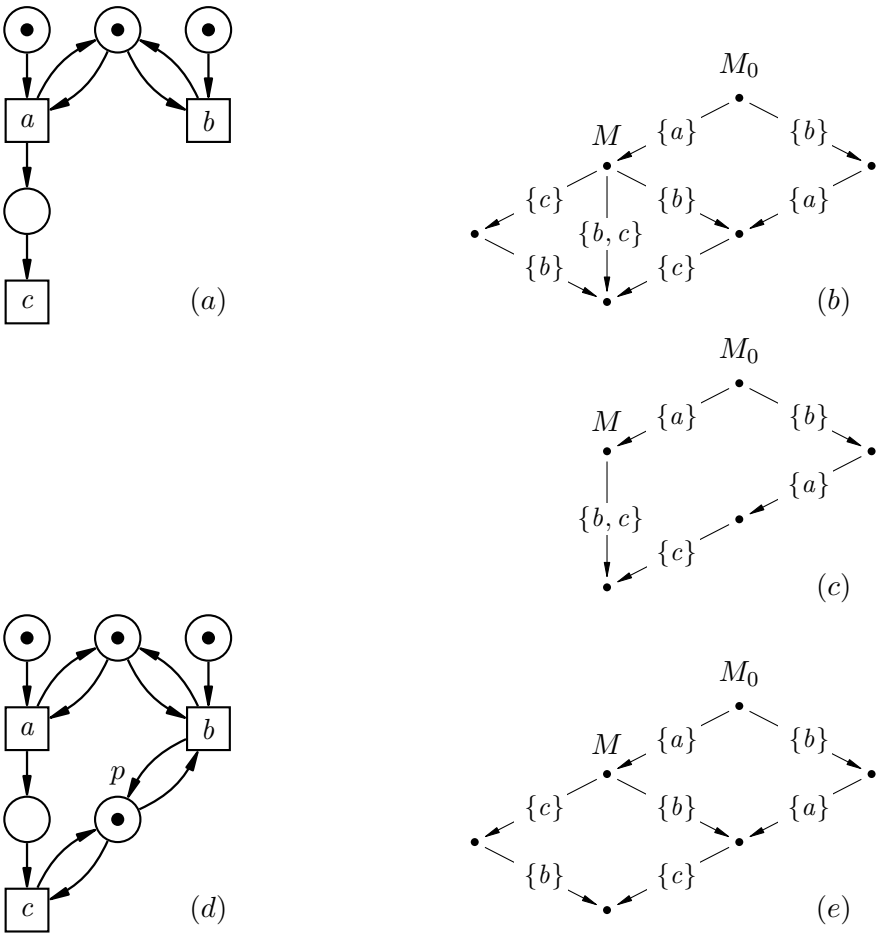


Figure 12: A safe persistent PT-net $\mathcal{N}$ (a); its concurrent reachability graph $CRG(\mathcal{N})$ (b); $CRG^{max}(\mathcal{N})$ obtained in the pruning procedure which does not satisfy $CRG^{max}(\mathcal{N}) \preccurlyeq^{local}_{pers} CRG(\mathcal{N})$ (c); a persistent and safe PT-net $\mathcal{N}' = \mathcal{N}_{b \leftrightarrow c}$ (d); and its concurrent reachability graph $CRG(\mathcal{N}') = CRG^{max}(\mathcal{N}')$ which trivially satisfies $CRG^{max}(\mathcal{N}') \preccurlyeq^{local}_{pers} CRG(\mathcal{N}')$ (and also $CRG^{max}(\mathcal{N}') \preccurlyeq^{global}_{pers} CRG(\mathcal{N}')$) (e).

Fact 5.2. Let $\mathcal{N}$ be a safe PT-net which is persistent according to Definition 3.1. Then $CRG(\mathcal{N}_{x \leftrightarrow y})$ is also persistent (according to Definition 3.1) and safe. ■

We can now propose a dynamic way of pruning embodied by the following algorithm:

while $\neg(CRG^{max}(\mathcal{N}) \preccurlyeq^{local}_{pers} CRG(\mathcal{N}))$ **do**
  choose M, α, β, γ *in* $CRG^{max}(\mathcal{N})$ *satisfying* (GB′) *with* $\alpha \subset \gamma$
  choose $x \in \alpha,\ y \in \gamma \setminus \alpha$
  $\mathcal{N} := \mathcal{N}_{x \leftrightarrow y}$

It follows from what we already demonstrated that the above algorithm always terminates and for the final

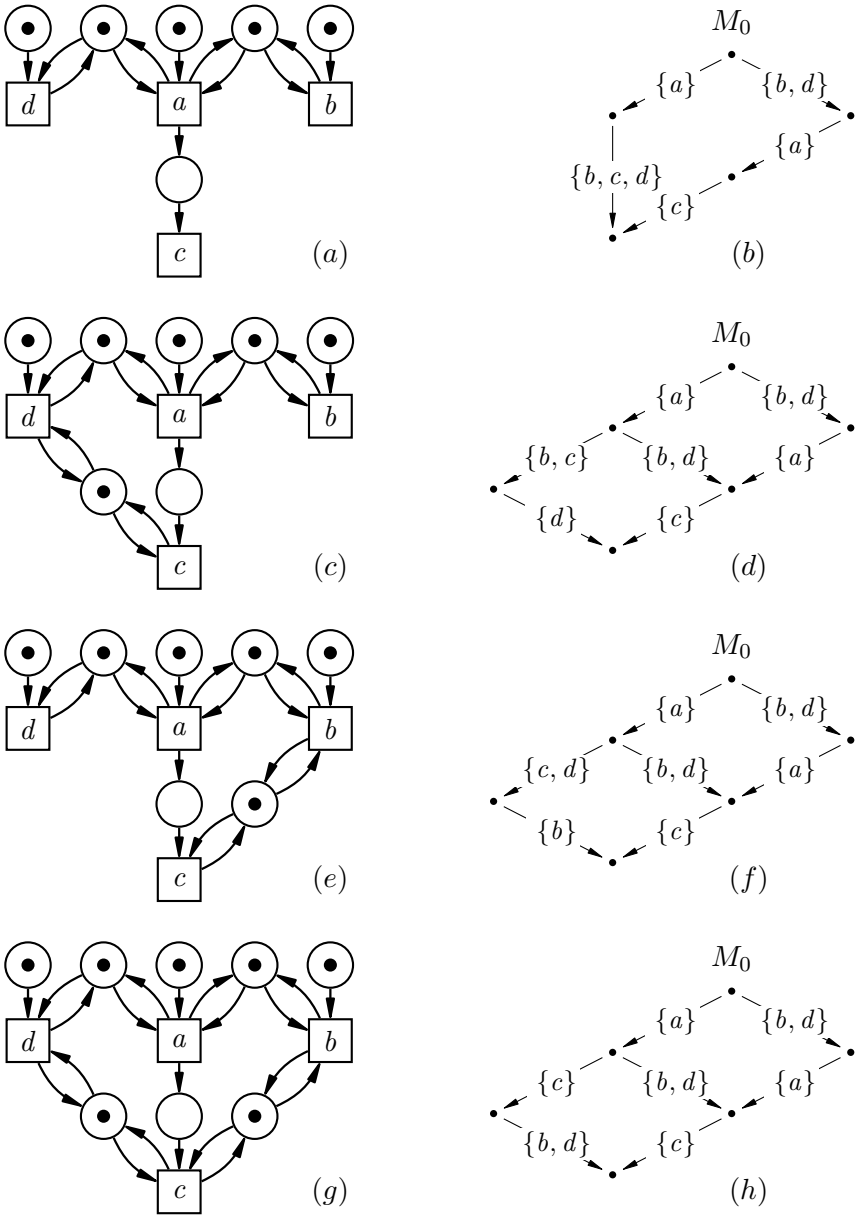


Figure 13: A safe persistent PT-net $\mathcal{N}$ (a); $CRG^{max}(\mathcal{N})$ not satisfying $CRG^{max}(\mathcal{N}) \preccurlyeq^{local}_{pers} CRG(\mathcal{N})$ (b); $\mathcal{N}' = \mathcal{N}_{d \leftrightarrow c}$ (c); $CRG^{max}(\mathcal{N}')$ (d); $\mathcal{N}'' = \mathcal{N}_{b \leftrightarrow c}$ (e); and $CRG^{max}(\mathcal{N}'')$ (f). Both $\mathcal{N}'$ and $\mathcal{N}''$ have been obtained as by-products of the successful runs of the pruning and modification algorithm. It is not possible to obtain in this way $\mathcal{N}'''$ in (g) even though $CRG^{max}(\mathcal{N}''') \preccurlyeq^{global}_{pers} CRG(\mathcal{N})$ (h).

PT-net $\mathcal{N}'$ we have $CRG^{max}(\mathcal{N}') \preccurlyeq^{local}_{pers} CRG(\mathcal{N}') \preccurlyeq CRG(\mathcal{N})$ and therefore $CRG^{max}(\mathcal{N}') \preccurlyeq^{local}_{pers} CRG(\mathcal{N})$. Since the algorithm is non-deterministic, we may try various strategies for choosing x and y. Figure 13 shows that different strategies may lead to different solutions.

In the context of GALS system design, pruning a system's concurrent specification enables a designer to generate a suitable behavioural model that offers the flexibility of asynchronous design coupled with the simplicity of synchronous design. The control circuit synthesised from such a specification will hence manage a mixture of sequential, parallel and concurrent request signals that execute the bundles of the pruned system while maintaining a functionally correct system flow. Specifically, a bundle deals with the execution of a single action or simultaneous execution of multiple actions in a single clock domain, while concurrent interactions between distinct bundles indicate synchronisation between different clock domains. For example, let us consider the Petri net model in Figure 14(a) which satisfies the temporal specification given in Figure 2(c). Actions a and b belong to different clock domains and have been specified to execute asynchronously. However, their completions must be synchronised before the execution of bundle $\{c, d\}$ which belongs to a third clock domain. Another GALS implementation of the same system is depicted behaviourally in Figure 14(b) and its derived Petri net model is shown in Figure 14(c). To give an idea of the practical implications of pruning in digital system design, the two pruned examples can be viewed as a classic case of trade-off between power and latency. A scenario where the latter implementation would be beneficial is the reduction of electromagnetic interference (EMI) which is caused by the simultaneous switching of circuit elements [15]. Here, if the concurrent execution of actions a and b leads to higher peak power as compared to the execution of bundle $\{c, d\}$, then the designer would want to order the bundles as indicated. This would be particularly intuitive when considering a complex system where many such blocks exist and a pruned behavioural specification is required to ensure low spectral energy peaks.

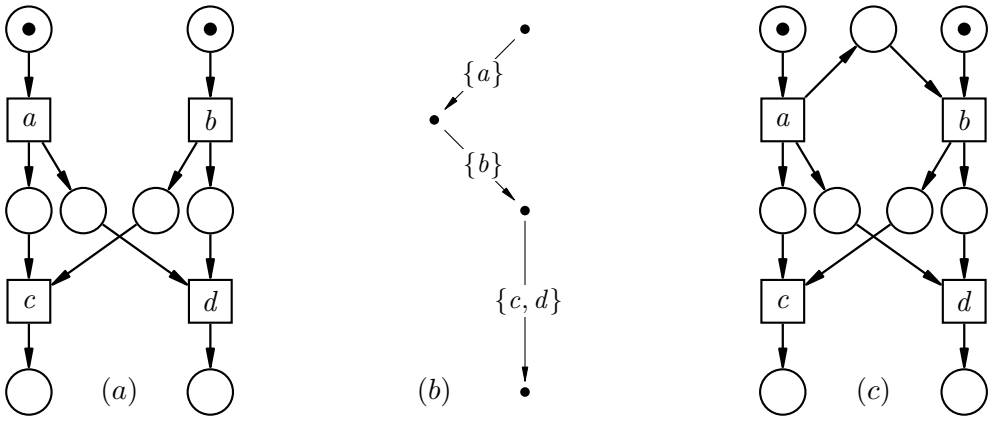


Figure 14: A PT-net generating the concurrent reachability graph of Figure 2(c) under the assumption that a and b belong to two different clock domains, while c and d belong to a third clock domain (a); and another GALS implementation of the same system (assuming the same clock domains as in (a)) both behaviourally (b) and as a derived PT-net (c).

It is essential that pruning is performed in a step persistent manner, according to the rules introduced in this paper. Each rule will ensure a distinct case of hazard-free step execution and also affect control circuit complexity differently. To briefly motivate how these rules affect GALS circuits, we discuss the case of LNI using the examples given in Figure 13. It should be noted that the enabling of a step, when

translated to circuit behaviour, results in switching the concerned request signal to a logical 1, triggering the execution of its comprising actions. The following paragraph explains how a non-monotonic pulse can arise if specific countermeasures are not considered in the circuit synthesis procedure.

The case of local inclusion of steps is compared with the case of local non-inclusion taking Figure 13(b) and Figure 13(h), respectively. In the former specification, the execution of $\{a\}$ and $\{b, d\}$ is concurrent but not simultaneous due to shared resources, as indicated in the Petri net model shown in Figure 13(a). If the arbitration of the shared resource favours executing action $\{a\}$, the step $\{b, d\}$ should get its resource after the completion of action $\{a\}$. However, owing to the given temporal specification, a new request signal pertaining to step $\{b, c, d\}$ will be generated, disabling the request of step $\{b, d\}$. This results in a non-monotonic pulse, as illustrated in Figure 15, which could lead to a hazardous computation as it is uncertain how the system will interpret the glitch. Such a glitch will not occur when executing the specification of Figure 13(h), as the action $\{b, d\}$ that was enabled does not get disabled until execution and hence is step persistent.

Now, if the former specification is a requirement for the design, only an arbitration between the request signals for the two steps $\{b, d\}$ and $\{b, c, d\}$ can ensure hazard freedom. This, however, complicates the control circuit synthesis due to the addition of an extra arbitration block which is not the case when implementing the latter. Step persistence therefore plays a vital role in the two main aspects of formal modelling of digital systems: in the verification of GALS systems to ensure freedom from hazard, and in the synthesis of GALS circuit implementation to ensure optimal control circuitry.

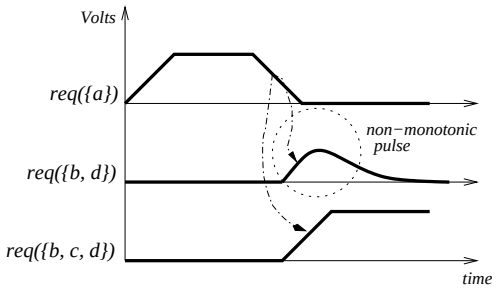


Figure 15: Hazardous switching in GALS step execution semantics.

6. Conclusions

In GALS, bundling is envisaged to reduce signal management, and could reduce the cost of scheduling and control, and improve system performance. The ideal way to model mixed synchronous-asynchronous systems is to start with a concurrent model that is persistent and fully asynchronous in behaviour. Then run several iterations that derive a combination of bundles that represents the temporal nature the designer requires. Careful selection of bundles is essential so that the pruned behaviour of the fully asynchronous model still exhibits some characteristics of its parent and is persistent. *Step persistence* is hence an important characteristic that will guarantee true persistent behaviour for mixed synchronous-asynchronous models.

In this paper we developed a pruning procedure for reachability graphs of persistent and safe nets.

This procedure constructs a step transition system that contains only bundles. The bundles in our algorithm represent maximally concurrent steps of the initial system that are persistent and satisfy non-inclusion constraints.

We hope that the reader have found the theory presented in this paper as a necessary first step in paving the way towards automating the design of GALS systems. Right now, we are not trying to answer how this theory can be applied in the scenarios of verification and synthesis mentioned in the introduction. This will be done in our subsequent papers, which will have to answer many new questions arising on the way, including, for example, what a rigorous metric for the complexity of bundle control is, how the notions of maximal steps (global and local) affect such a complexity, or what the different forms of step persistence (type-A, type-B and type-C) imply in terms of hazard-avoidance in the system.

A move into the realm of step based execution semantics creates a wealth of new fundamental problems and intriguing questions, some of which have been addressed in [10, 14]. In particular, there are different ways in which the standard notion of persistence could be lifted from the level of sequential semantics to the level of step semantics. For example, if part of an enabled step has been executed by another step, should we insist on the whole delayed step to be still enabled, or just its remaining part? Moreover, one may consider steps which are persistent and cannot be disabled by other steps, as well as steps which are nonviolent [1, 2, 14] and cannot disable other steps.

In the future we intend to investigate other possible pruning algorithms, weakening our constraints and allowing the initial system's behaviour to be given by a net that is not necessarily persistent. Furthermore, we plan to allow in the algorithms the choice of non-maximal bundles in certain cases. For example, input signals are usually behaving in fully asynchronous way and should not be bundled.

This is the place, where nonviolence should start to play a foreground role. Note that persistence and nonviolence bring distinct advantages to the design process. In particular, one may start from a persistent skeleton and then perform a series of nonviolent additions. Alternatively, and as we do it in this paper, one may perform a series of persistent deletions from a persistent skeleton. In a way, persistence is analytical as persistent deletions require careful considerations of already existing steps, whereas nonviolent additions trigger a constructive process. Another potential advantage of nonviolence is that under realistic progress conditions, one might first execute nonviolent transitions (phantom transitions) without disabling any other transitions.

Acknowledgments

We are grateful to the reviewers for their useful comments and suggestions. This research was supported by the National Science Center under the grant No.2013/09/D/ST6/03928, the 973 Program Grant 2010CB328102, NSFC Grant 61133001, and the EPSRC GAELS and UNCOVER projects.

References

[1] Barylska, K., Ochmański, E.: Levels of persistency in place/transition nets. Fundamenta Informaticae 93 (2009) 33–43.

[2] Barylska, K., Mikulski, Ł., Ochmański, E.: On persistent reachability in Petri nets. Information and Computation 223 (2013) 67–77.

[3] Best, E., Darondeau, Ph.: Decomposition theorem for bounded persistent Petri nets. ICATPN'08, Lecture Notes in Computer Science 5062, Springer (2008) 33–51.

[4] Best, E., Darondeau, Ph.: Separability in persistent Petri nets. ICATPN'10, Lecture Notes in Computer Science 6128 (2010) 246–266.

[5] Chapiro, D.M.: Globally-asynchronous locally-synchronous systems. PhD Thesis, Stanford University (1984).

[6] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Logic synthesis for asynchronous controllers and interfaces. Springer Series in Advanced Microelectronics 8, Springer-Verlag (2002).

[7] Dasgupta, S., Potop-Butucaru, D., Caillaud, B., Yakovlev, A.: Moving from weakly endochronous systems to delay-insensitive circuits. Electronic Notes in Theoretical Computer Science 146 (2006) 81–103

[8] Dasgupta, S., Yakovlev, A.: Desynchronisation technique using Petri nets. Electronic Notes in Theoretical Computer Science 245 (2009) 51–67.

[9] Davis, A., Nowick, S.M.: An introduction to asynchronous circuit design. The Encyclopedia of Computer Science and Technology (1997).

[10] Fernandes, J., Koutny, M., Pietkiewicz-Koutny, M., Sokolov, D., Yakovlev, A.: Step persistence in the design of GALS systems. ICATPN'13, Lecture Notes in Computer Science 7927, Springer (2013) 190–209.

[11] Gurkaynak, F., Oetiker, S., Kaeslin, H., Felber, N., Fichtner, W.: GALS at ETH Zurich: Success or failure? Proc. of the 12th IEEE International Symposium on Asynchronous Circuits and Systems (2006).

[12] Iyer, A., Marculescu, D.: Power and performance evaluation of globally asynchronous locally synchronous processors. 29th International Symposium on Computer Architecture, IEEE Computer Society (2002) 158–168.

[13] Keller, R.: A fundamental theorem of asynchronous parallel computation. Lecture Notes in Computer Science 24 (1975) 102–112.

[14] Koutny, M., Mikulski, Ł., Pietkiewicz-Koutny, M.: A taxonomy of persistent and nonviolent steps. ICATPN'13, Lecture Notes in Computer Science 7927, Springer (2013) 210-229.

[15] Król, T., Krstić, M., Fan, X., Grass, E.: Modeling and reducing EMI in GALS and synchronous systems. PATMOS'09, Lecture Notes in Computer Science 5953, Springer (2009) 146–155.

[16] Landweber, L.H., Robertson, E.L.: Properties of conflict-free and persistent Petri nets. JACM 25 (1978) 352–364.

[17] Muller, D.E., Bartky, W.S.: A theory of asynchronous circuits. Proceedings of an International Symposium on the Theory of Switching, Harvard University Press (1959) 204-243.

[18] Myers, C.: Asynchronous circuit design. Wiley (2004).

[19] Sparso, J., Furber, S.: Principles of asynchronous circuit design: a systems perspective. Kluwer Academic Publishers (2001).

[20] Stevens, K.S., Gebhardt, D., You, J., Xu, Y., Vij, V., Das, S., Desai, K.: The future of formal methods and GALS design. Electronic Notes in Theoretical Computer Science 245 (2009) 115–134.

[21] Yakovlev, A.: Designing self-timed systems. VLSI SYSTEM DESIGN VI (1985) 70–90.

[22] Yakovlev, A., Koelmans, A., Semenov, A., Kinniment, D.: Modelling, analysis and synthesis of asynchronous control circuits using Petri nets. INTEGRATION, the VLSI Journal 21 (1996) 143–170.