A symbolic algorithm for the synthesis of bounded Petri nets*

J. Carmona[1], J. Cortadella[1], M. Kishinevsky[2], A. Kondratyev[3], L. Lavagno[4], and A. Yakovlev[5]

[1] Universitat Politècnica de Catalunya, Spain
[2] Intel Corporation, USA
[3] Cadence Berkeley Laboratories, USA
[4] Politecnico di Torino, Italy
[5] Newcastle University, UK

Abstract. This paper presents an algorithm for the synthesis of bounded Petri nets from transition systems. A bounded Petri net is always provided in case it exists. Otherwise, the events are split into several transitions to guarantee the synthesis of a Petri net with bisimilar behavior. The algorithm uses symbolic representations of multisets of states to efficiently generate all the minimal regions. The algorithm has been implemented in a tool. Experimental results show a significant net reduction when compared with approaches for the synthesis of safe Petri nets.

1 Introduction

The problem of Petri net synthesis consists of building a Petri net that has a behavior equivalent to a given transition system. The problem was first addressed by Ehrenfeucht and Rozenberg [ER90] introducing *regions* to model the sets of states that characterize marked places. Mukund [Muk92] extended the concept of region to synthesize nets with weighted arcs.

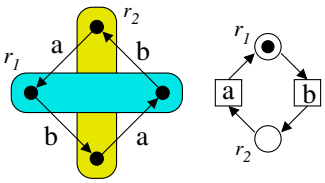
Different variations of the synthesis problem have been studied in the past [Dar07]. Most of the efforts have been devoted to the decidability problem, i.e. questioning about the existence of a Petri net with a specified behavior. An exception is in [BBD95] where polynomial algorithms for the synthesis of bounded nets were presented. These methods have been implemented in the tool SYNET [Cai02].

Desel and Reisig [DR96] reduced the synthesis problem to the calculation of the subset of *minimal regions*. In [HKT95,HKT96] the classical trace model by Mazurkiewicz [Maz87] was extended to describe the behavior of general Petri nets.

* Work of J. Carmona and J. Cortadella has been supported by the project FORMALISM (TIN2007-66523), and a grant by Intel Corporation. Work of A. Yakovlev was supported by EPSRC, Grants EP/D053064/1 and EP/E044662/1.

Carmona, J. [et al.]. A symbolic algorithm for the synthesis of bounded Petri nets. A: International Conference on Applications and Theory of Petri Nets. "Applications and Theory of Petri Nets: 29th International Conference, PETRI NETS 2008, Xi'an, China, June 23-27, 2008: proceedings". Springer, 2008, p. 92-111.
The final authenticated version is available online at https://doi.org/10.1007/978-3-540-68746-7_10

In most of the previous approaches, two major constraints are imposed: (1) any event must be represented by *only one transition*, and (2) the reachability graph of the Petri net must be *isomorphic* to the initially given transition system. The approach presented in [CKLY98] relaxed the previous conditions. The condition of isomorphism was replaced by *bisimilarity* [Mil89] and, as a result, the classical *separation axioms* did not have to hold for pairs of bisimilar states. Additionally, the Petri net was allowed to have multiple transitions with the same label (event). This approach was only applicable to safe Petri nets. The figure shows a transition system where the separation axioms do not hold, but a Petri net with bisimilar behavior can be found with the methods described in [CKLY98].



1.1 Motivation and contributions

There are several areas of interest for the synthesis of Petri nets. One of them is *visualization* [VPWJ07]. Understanding the behavior of large concurrent systems such as business processes [BDLS07] or asynchronous circuits [CKLY98] is a complex task that can be facilitated by visualizing the causality and concurrency relations of their events. In these cases, the non-existence of a Petri net should not be the cause that prevents visualization.

Another interesting area of application is *direct synthesis*, which consists of implementing concurrent systems by representing them as Petri nets and mapping the places and transitions into software of hardware realizations of these objects (e.g. [SBY07]). Finding succinct representations of concurrent behaviors contributes to derive efficient implementations.

On the other hand, the state spaces of such concurrent systems do not always have an explicit representation. Instead, symbolic representations are often used. This is the case of the transition relations used to verify temporal properties [CGP00] or the gate netlists used to represent circuits. The behavior is implicitly represented by the reachable states obtained from these representations.

This paper provides an efficient synthesis approach for concurrent systems. An algorithm for bounded Petri nets synthesis based on the theory of general regions is presented. Starting from the algorithms for synthesizing safe Petri Nets in [CKLY98], the theory and algorithms are extended by generalizing the notion of excitation closure from sets of states to multisets of states. The extension covers the case of the k-bounded Petri nets with weighted arcs. The paper also proposes heuristics to handle transition systems which do not satisfy the notion of excitation closure and hence cannot be modeled with general Petri nets with uniquely labelled transitions. In this case, methods for splitting events allow to generate Petri Nets with multiple occurrences of the same original label. In summary, the main features of the approach are:

– The synthesis of weighted Petri nets.

– The use of symbolic methods based on BDDs to explore large state spaces.
– Efficient heuristics for event splitting.

1.2 Two illustrative examples

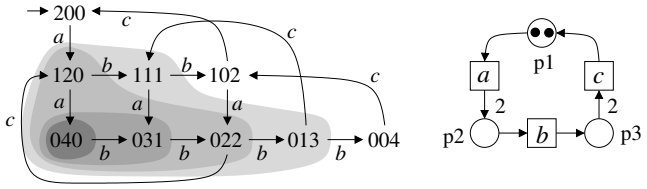


Fig. 1: A transition system and an equivalent bounded Petri net.

Figure 1 depicts a finite transition system with 9 states and 3 events. After synthesis, the Petri net at the right is obtained. Each state has a 3-digit label that corresponds to the marking of places p_1, p_2 and p_3 of the Petri net, respectively. The shadowed states represent the general region that characterizes place p_2. Each grey tone represents a different multiplicity of the state (4 for the darkest and 1 for the lightest). Each event has a gradient with respect to the region (+2 for a, -1 for b and 0 for c). The gradient indicates how the event changes the multiplicity of the state after firing. For the same example, the equivalent *safe* Petri net generated by `petrify` [CKLY98] has 5 places and 10 transitions.

Another example is shown in Fig. 2. The transition system models a behavior with OR-causality, i.e. the event c can be fire as soon as a or b have fired. The net model is much simpler and intuitive if bounded nets are used. Instead, the model with a safe Petri net needs to represent the events a and b with multiple transitions.

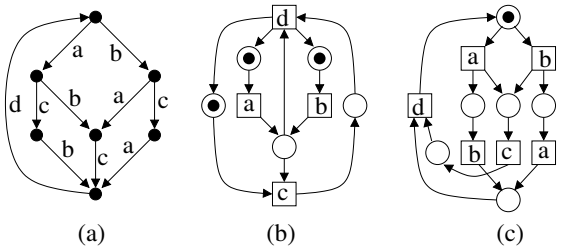


Fig. 2: (a) Transition system, (b) 2-bounded Petri net, (c) safe Petri net.

The algorithm presented in this paper is based on an efficient manipulation of multisets to explore the minimal general regions of a finite transition system.

2 Background

2.1 Petri nets and finite transition systems

Definition 1. *A Petri net is a tuple (P, T, W, M_0) where P and T represent finite sets of places and transitions, respectively, and $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is the weighted flow relation. The initial marking $M_0 \in \mathbb{N}^{|P|}$ defines the initial state of the system.*

Definition 2 (Transition system). *A* transition system *(or* TS*) is a tuple (S, Σ, E, s_{in}), where S is a set of* states*; Σ is an alphabet of* actions*, such that $S \cap \Sigma = \emptyset$; $E \subseteq S \times \Sigma \times S$ is a set of* (labelled) transitions*; and s_{in} is the* initial state.

Let $\mathsf{TS} = (S, \Sigma, E, s_{in})$ be a transition system. We consider connected TSs that satisfy the following axioms:

- S and E are finite sets.
- Every event has an occurrence: $\forall e \in \Sigma : \exists (s, e, s') \in E$;
- Every state is reachable from the initial state: $\forall s \in S : s_{in} \xrightarrow{*} s$.

A state without outgoing arcs is called *deadlock* state.

2.2 Multisets

The following definitions establish the necessary background to understand the concept of region, introduced in Section 2.3. A region is a multiset where additional conditions hold. We start by introducing the multiset terminology.

Definition 3 (Multiset). *Given a set S, a multiset r of S is a mapping $r : S \longrightarrow \mathbb{N}$. We will also use a set notation for multisets. For example, let $S = \{s_1, s_2, s_3, s_4\}$, then a multiset $r = \{s_1^3, s_2^2, s_3\}$ corresponds to the following mapping $r(s_1) = 3, r(s_2) = 2, r(s_3) = 1, r(s_4) = 0$.*

Henceforth we will assume r, r_1 and r_2 to be multisets of a set S.

Definition 4 (Support of a multiset). *The support of a multiset r is defined as*

$$\mathrm{supp}(r) = \{s \in S \mid r(s) > 0\}$$

Definition 5 (Power of a multiset). *The power of a multiset r, denoted by $r^\diamond$, is defined as*

$$r^\diamond = \max_{s \in S} r(s)$$

For instance, for the multiset $r = \{s_1^3, s_2^2, s_3\}$, $r^\diamond = 3$.

Definition 6 (Trivial multisets). *A multiset r is said to be trivial if $r(s) = r(s')$ for all $s, s' \in S$. The trivial multisets will be denoted by $\mathbf{0}$, $\mathbf{1}$, ..., $\mathbf{K}$ when $r(s) = 0$, $r(s) = 1$, ..., $r(s) = k$, for every $s \in S$, respectively.*

Definition 7 (k-bounded multiset). *A multiset r is k-bounded if for all $s \in S : r(s) \leq k$.*

Definition 8 (Union, intersection and difference of multisets). *The union, intersection and difference of two multisets r_1 and r_2 are defined as follows:*

$$(r_1 \cup r_2)(s) = \max(r_1(s), r_2(s))$$
$$(r_1 \cap r_2)(s) = \min(r_1(s), r_2(s))$$
$$(r_1 - r_2)(s) = \max(0, r_1(s) - r_2(s))$$

Definition 9 (Subset of a multiset). *A multiset r_1 is a subset of a multiset r_2 ($r_1 \subseteq r_2$) if*

$$\forall s \in S \ : \ r_1(s) \leq r_2(s)$$

As usual, we will denote by $r_1 \subset r_2$ the fact that $r_1 \subseteq r_2$ and $r_1 \neq r_2$.

Definition 10 (k-topset of a multiset). *The k-topset of a multiset r, denoted by $\top_k(r)$, is defined as follows:*

$$\top_k(r)(s) = \begin{cases} r(s) & \text{if } r(s) \geq k \\ 0 & \text{otherwise} \end{cases}$$

A multiset r_1 is a topset of r_2 if there exists some k for which $r_1 = \top_k(r_2)$.

Examples. The multiset $\{s_1^3, s_3\}$ is a subset of $\{s_1^3, s_2^2, s_3\}$, but it is not a topset. The multisets $\{s_1^3, s_2^2\}$ and $\{s_1^3\}$ are the 2- and 3-topsets of $\{s_1^3, s_2^2, s_3\}$, respectively. As it will be shown in Section 4, k-topsets are the main objects to look at when constructing the (weighted) flow relation for Petri net synthesis.

Property 1 (Partial order of multisets). The relation $\subseteq$ (subset) on the set of multisets of S is a partial order.

Property 2 (The lattice of k-bounded multisets). The set of k-bounded multisets of a set S with the relation $\subseteq$ is a lattice. The meet and join operations are the intersection and union of multisets respectively. The least and greatest elements are $\mathbf{0}$ and $\mathbf{K}$ respectively.

2.3 General regions

Let $\mathsf{TS} = (S, \Sigma, E, s_{in})$ be a transition system. In this section, we will consider multisets of the set S.

Definition 11 (Gradient of a transition). *Given a multiset r, the gradient of a transition (s, e, s') is defined as*

$$\Delta_r(s, e, s') = r(s') - r(s)$$

An event e is said to have a non-constant gradient in r if there are two transitions (s_1, e, s_1') and (s_2, e, s_2') such that

$$r(s_1') - r(s_1) \neq r(s_2') - r(s_2)$$

Definition 12 (Region). *A multiset r is a region if all events have a constant gradient in r.*

The original notion of region from [ER90] was restricted to subsets of S, i.e. events could only have gradients in $\{-1, 0, +1\}$.

Definition 13 (Gradient of an event). *Given a region r and an event e with $(s, e, s') \in E$, the gradient of e in r is defined as*

$$\Delta_r(e) = r(s') - r(s)$$

Definition 14 (Minimal region). *A region r is minimal if there is no other region $r' \neq \mathbf{0}$ such that $r' \subset r$.*

Note that the trivial region $\mathbf{0}$ is not considered to be minimal.

Theorem 1. *Let r be a region such that $\mathbf{1} \subset r$. Then r is not a minimal region.*

Proof. Trivial. A smaller region can be obtained by subtracting 1 from each $r(s)$. □

Definition 15 (Excitation and switching regions[6]). *The excitation region of an event e, $\mathsf{ER}(e)$, is the set of states in which e is enabled, i.e.*

$$\mathsf{ER}(e) = \{s \mid \exists s' : (s, e, s') \in E\}$$

The switching region of an event e, $\mathsf{SR}(e)$, is the set of states reachable from $\mathsf{ER}(e)$ after having fired e, i.e.

$$\mathsf{SR}(e) = \{s \mid \exists s' : (s', e, s) \in E\}$$

For convenience, $\mathsf{ER}(e)$ and $\mathsf{SR}(e)$ will be also considered as multisets of states when necessary.

Definition 16 (Pre- and post-regions). *A region r is a pre-region of e if $\mathsf{ER}(e) \subseteq r$. A region r is a post-region of e if $\mathsf{SR}(e) \subseteq r$. The sets of pre- and post-regions of an event e are denoted by ${}^\circ e$ and e° respectively.*

Note that a region r can be a pre-region and a post-region of the same event in case $\mathsf{ER}(e) \cup \mathsf{SR}(e) \subseteq r$. The behavior modeled in this situation can be seen as a self-loop in a Petri net.

[6] Excitation and switching regions are not regions in the terms of Definition 12. They correspond to the set of states in which an event is enabled or just fired, correspondingly. The terms are used due to historical reasons.

Properties of regions

Property 3. Let $\mathsf{TS} = (S, \Sigma, E, s_{in})$ be a transition system without deadlock states. Then, for any region r there is an event e for which $\Delta_r(e) \leq 0$.

Proof. By contradiction. Assume that $\Delta_r(e) > 0$ for all events. Then for all arcs (s, e, s') we have that $r(s') > r(s)$. Since TS has no deadlock states and S is finite, there is at least one cycle $s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n \rightarrow s_0$. This would result in $r(s_0) > r(s_0)$, which is a contradiction. □

Property 4. Let $\mathsf{TS} = (S, \Sigma, E, s_{in})$ be a transition system in which there is a transition $(s, e', s_{in}) \in E$. Then, for any region r there is an event e for which $\Delta_r(e) \geq 0$.

Proof. By contradiction. Assume that $\Delta_r(e) < 0$ for all events. Then for all arcs (s, e, s') we have that $r(s') < r(s)$. Since there is $(s, e', s_{in}) \in E$, there is at least one cycle $s_{in} \rightarrow s_1 \rightarrow \ldots \rightarrow s_n = s \rightarrow s_{in}$. This would result in $r(s_{in}) < r(s_{in})$, which is a contradiction. □

Property 5. Let $\mathsf{TS} = (S, \Sigma, E, s_{in})$ be a transition system without deadlock states. Then, any region $r \neq \emptyset$ is the pre-region of some event e.

Proof. Property 3 guarantees the existence of an event e such that $\Delta_r(e) \leq 0$. If $\Delta_r(e) < 0$, then every state in $\mathsf{ER}(e)$ must be in the support of r and the claim holds. If every event e fulfilling Property 3 satisfies $\Delta_r(e) = 0$, then $supp(r) = S$: if the contrary is assumed, since the transition system is deadlock-free and $r \neq \emptyset$, the states in the support of r must be connected to some states out of r. However, if every event has non-negative gradient in r, then in this situation r must contain all the states. □

Property 6. Let $\mathsf{TS} = (S, \Sigma, E, s_{in})$ be a transition system. Then, any region $r \neq \emptyset$ is the pre-region or the post-region of some event e.

Proof. A similar reasoning of the proof of Property 5 can be applied here, but using Properties 3 and 4. □

2.4 Excitation-closed TSs

This section defines a specific class of transition systems, called *excitation-closed*, for which the synthesis approach presented in this paper guarantees that the Petri net obtained has a reachability graph bisimilar to the initial transition system.

Definition 17 (Enabling topset). *The set of smallest enabling topsets of an event e is denoted by ${}^\star e$ and defined as follows:*

$${}^\star e = \{q \mid \exists r \in {}^\circ e, k > 0 : \; q = \top_k(r) \; \wedge \; \mathsf{ER}(e) \subseteq \top_k(r) \; \wedge \; \mathsf{ER}(e) \not\subseteq \top_{k+1}(r)\}$$

Intuitively, q belongs to ${}^\star e$ if it is the topset of a pre-region r of e and there is no larger topset that includes $\mathsf{ER}(e)$.

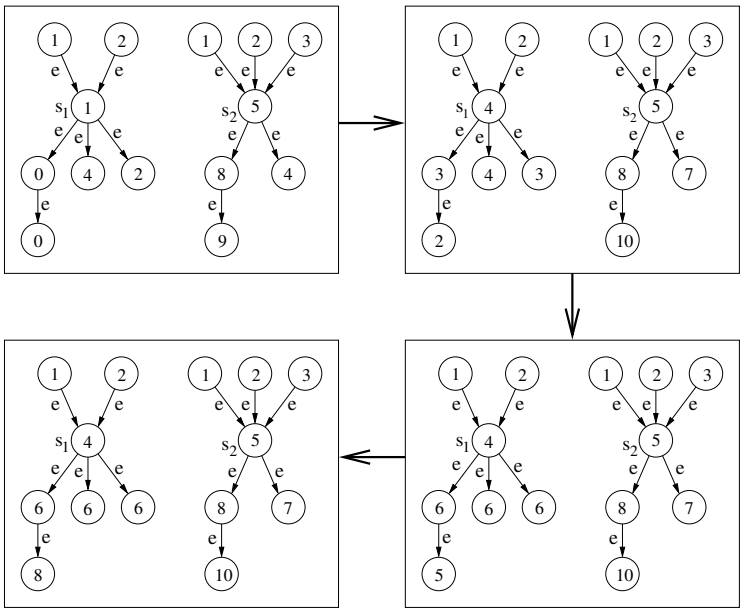


Fig. 3: Successive calculations of $\sqcap^2(r, e)$.

Definition 18 (ECTS). *A* TS *is* excitation closed *if it satisfies the following two properties:*

1. *Excitation closure. For each event* e

$$\bigcap_{q \in {}^\star e} \mathrm{supp}(q) = \mathsf{ER}(e)$$

2. *Event effectiveness. For each event* e, ${}^\circ e \neq \emptyset$

3 Generation of minimal regions

Now we are ready to describe an algorithm to generate the set of all k-bounded minimal regions from a given transition system. Informally, the generation of minimal regions is based on Property 6, that states that any region is either a pre-region or post-region of some event. Therefore the exploration of regions starts by considering the ER and SR of every event, and expands those sets that violate the region condition. Provided that only minimal expansions are considered at each step of the algorithm, the generation of all the minimal regions is guaranteed. Hence the notion of multiset expansion is crucial in this paper.

Multisets are expanded with the aim of ensuring constant gradient for every event. Formally, given a multiset r and an event e with non-constant gradient, the following definitions characterize the set of regions that include r.

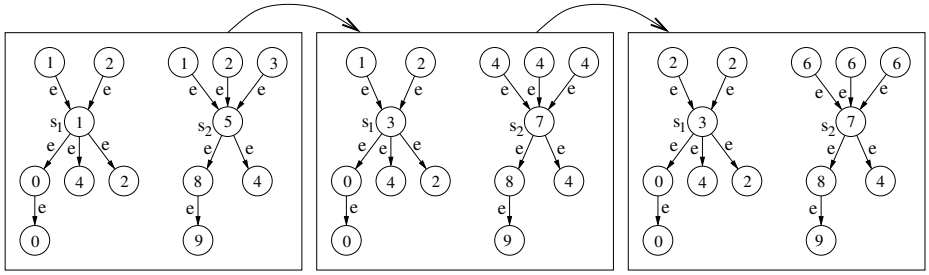


Fig. 4: Successive calculations of $\sqcap_1(r, e)$.

Definition 19. *Let $r \neq \mathbf{0}$ be a multiset. We define*

$$\mathcal{R}_g(r, e) = \{r' \supseteq r | r' \text{ is a region and } \Delta_{r'}(e) \leq g\}$$
$$\mathcal{R}^g(r, e) = \{r' \supseteq r | r' \text{ is a region and } \Delta_{r'}(e) \geq g\}$$

$\mathcal{R}_g(r, e)$ is the set of all regions larger than r in which the gradient of e is smaller than or equal to g. Similar for $\mathcal{R}^g(r, e)$ and the gradient of e greater than or equal to g. Notice that in this definition a gradient g is used to partition the set of regions including r into two classes. This binary partition is the basis for the calculation of minimal k-bounded regions that will be presented at the end of this section.

To expand a multiset in order to convert it into a region, it is necessary to know the lower bound needed for the increase in each state in order to satisfy a given gradient constraint. The next functions provide these lower bounds:

Definition 20. *Given a multiset r, a state s and an event e, the following δ functions are defined*[7]*:*

$$\delta_g(r, e, s) = \max(0, \max_{(s,e,s') \in E}(r(s') - r(s) - g))$$
$$\delta^g(r, e, s) = \max(0, \max_{(s',e,s) \in E}(r(s') - r(s) + g))$$

Informally, δ_g denotes a lower bound for the increase of $r(s)$, taking into account the arcs leaving from s, to force $\Delta_{r'}(e) \leq g$ in some region r' larger than r. Similarly, δ^g denotes a lower bound taking into account the arcs arriving at s, to force $\Delta_{r'}(e) \geq g$. Let us use Figure 3 to illustrate this concept. In the figure each state is labeled with $r(s)$. For the states s_1 and s_2 in the top-left figure, we have:

$$\delta^2(r, e, s_1) = 3 \qquad \delta^2(r, e, s_2) = 0$$

$\delta^2(r, e, s_1)$ is determined by the arc $\boxed{2} \xrightarrow{e} \boxed{1}$ and indicates that $r'(s_1) \geq 4$ in case we seek a region $r' \supset r$ with $\Delta_{r'}(e) \geq 2$. Analogously, Figure 4 illustrates the symmetrical concept. For the states s_1 and s_2 in the leftmost figure we have:

[7] For convenience, we consider $\max_{x \in D} P(x) = 0$ when the domain D is empty.

$$\delta_1(r, e, s_1) = 2 \qquad \delta_1(r, e, s_2) = 2$$

$\delta_1(r, e, s_1)$ is determined by the arc $\boxed{1} \xrightarrow{e} \boxed{4}$, indicating that $r'(s_1) \geq 3$ for $\Delta_{r'}(e) \leq 1$. Similarly, $\delta_1(r, e, s_2)$ is determined by the arc $\boxed{5} \xrightarrow{e} \boxed{8}$.

Definition 21. *Given a multiset r and an event e, the multisets $\sqcap_g(r, e)$ and $\sqcap^g(r, e)$ are defined as follows:*

$$\sqcap_g(r, e)(s) = r(s) + \delta_g(r, e, s)$$
$$\sqcap^g(r, e)(s) = r(s) + \delta^g(r, e, s)$$

Intuitively, $\sqcap_g(r, e)$ is a safe move towards growing r and obtaining all regions r' with $\Delta_r(e) \leq g$. Similarly, $\sqcap^g(r, e)$ for those regions with $\Delta_r(e) \geq g$. It is easy to see that $\sqcap_g(r, e)$ and $\sqcap^g(r, e)$ always derive multisets larger than r. The successive calculations of $\sqcap^g(r, e)$ and $\sqcap_g(r, e)$ are illustrated in Figures 3 and 4, respectively.

Theorem 2 (Expansion on events).

(a) Let $r \neq \mathbf{0}$ be a multiset and e an event such that there exists some (s, e, s') with $r(s') - r(s) > g$. The following hold:
1. $r \subset \sqcap_g(r, e)$
2. $\mathcal{R}_g(r, e) = \mathcal{R}_g(\sqcap_g(r, e), e)$

(b) Let $r \neq \mathbf{0}$ be a multiset and e an event such that there exists some (s, e, s') with $r(s') - r(s) < g$. The following hold:
1. $r \subset \sqcap^g(r, e)$
2. $\mathcal{R}^g(r, e) = \mathcal{R}^g(\sqcap^g(r, e), e)$

Proof. (We prove item (a), item (b) is similar.)
(a.1) Given the event e, for all states s, s' with (s, e, s') either (i) $r(s') - r(s) \leq g$ or (ii) $r(s') - r(s) > g$. In situation (i), the following equality holds: $r(s) = \sqcap_g(r, e)(s)$ because $\delta_g(r, e, s) = 0$ by Definition 20. In situation (ii), $\delta_g(r, e, s) > 0$, and therefore $r(s) < \sqcap_g(r, e)(s)$. Given that the rest of states without outgoing arcs labeled e fulfill also $r(s) = \sqcap_g(r, e)(s)$, and because there exists at least one transition satisfying (ii), the claim holds.
(a.2) To obtain $\mathcal{R}_g(r, e)$ it is necessary to guarantee $\Delta_{r'}(e) \leq g$ for each region $r' \supseteq r$. Given (s, e, s') with $r(s') - r(s) > g$, two possibilities can induce a gradient lower than g, e.g decreasing $r(s')$ or increasing $r(s)$, but only the latter leads to a multiset with r as a subset. □

Figure 5 presents an algorithm for the calculation of all minimal k-bounded regions. It is based on a dynamic programming approach that, starting from a multiset, generates an exploration tree in which an event with non-constant gradient is chosen at each node. All possible gradients for that event are explored by means of a binary search. Dynamic programming with memoization avoids the exploration of multiple instances of the same node. The final step of the algorithm (lines `16-17`) removes all those multisets that are neither regions nor minimal regions that have been generated during the exploration.

```
  generate_minimal_regions (TS,k) {
1:  R = ∅; /* set of explored multisets */
2:  P = {ER(e) | e ∈ E} ∪ {SR(e) | e ∈ E};
3:  while (P ≠ ∅) /* multisets pending for exploration */
4:    r = remove_one_element (P);
5:    if (r ∉ R) /* dynamic programming with memoization */
6:      R = R ∪ {r};
7:      if (r is not a region)
8:        e = choose_event_with_non_constant_gradient (r);
9:        (g_min, g_max) = ''minimum and maximum gradients of e in r'';
10:       g = ⌊(g_min + g_max)/2⌋; /* gradient for binary search */;
11:       r_1 = ⊓_g(r, e); if ((r_1^◇ ≤ k) ∧ (1 ⊄ r_1)) P = P ∪ {r_1} endif;
12:       r_2 = ⊓^{g+1}(r, e); if ((r_2^◇ ≤ k) ∧ (1 ⊄ r_2)) P = P ∪ {r_2} endif;
13:     endif
14:   endif
15  endwhile;
16: R = R \ {r | r is not a region}; /* Keep only regions */
17: R = R \ {r | ∃r' ∈ R : r' ⊂ r}; /* Keep only minimal regions */
  }
```

Fig. 5: Algorithm for the generation of all k-bounded minimal regions.

Theorem 3. *The algorithm* generate_minimal_regions *in Figure 5 calculates all k-bounded minimal regions.*

Proof. The proof is based on the following facts:

1. All minimal regions are a pre- or a post-region of some event (property 6). Any pre- (post-) region of an event is larger than its ER (SR). Line 2 of the algorithm puts all seeds for exploration in P. These seeds are the ERs and SRs of all events.
2. Each r that is not a region is enlarged by $\sqcap_g(r,e)$ and $\sqcap^{g+1}(r,e)$ for some event e with non-constant gradient. Given that $g = \lfloor (g_{min}+g_{max})/2 \rfloor$, there is always some transition $s_1 \xrightarrow{e} s_2$ such that $r(s_2) - r(s_1) = g_{max} > g$ and some transition $s_3 \xrightarrow{e} s_4$ such that $r(s_4) - r(s_3) = g_{min} < g+1$. Therefore, the conditions for theorem 2 hold. By exploring $\sqcap_g(r,e)$ and $\sqcap^{g+1}(r,e)$, no minimal regions are missed.
3. The algorithm halts since the set of k-bounded multisets with $\subseteq$ is a lattice and the multisets derived at each level of the tree are larger than their predecessors. Thus, the exploration will halt at those nodes in which the power of the multiset is larger than k (lines 11-12). The condition $(1 \not\subset r')$ in lines 11-12 improves the efficiency of the search (see theorem 1). □

For the case of safe Petri nets, line 9 of the algorithm always gives $g = g_{min} = 0$ and $g_{max} = 1$. The calculation of $\sqcap_0(r,e)$ and $\sqcap^1(r,e)$ with the constraints $r_1^\diamond, r_2^\diamond \leq 1$ is equivalent to the expansion of sets of states presented in lemma 4.2 of [CKLY98].

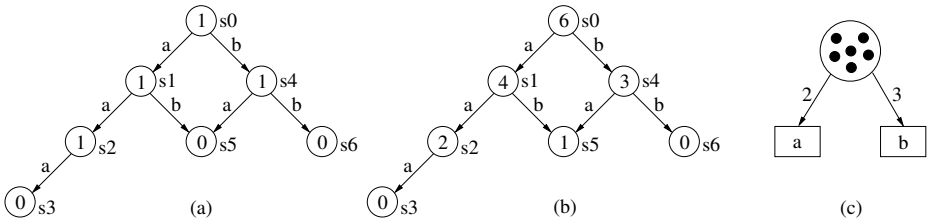


Fig. 6: Generation of minimal region: (a) $\mathsf{ER}(a)$, (b) final region after the exploration shown in table 1, (c) equivalent Petri net.

	$r_i(s)$							illegal	chosen			
r_i	s_0	s_1	s_2	s_3	s_4	s_5	s_6	events	event	g_{min}	g_{max}	g
$r_1 = \mathsf{ER}(a)$	1	1	1	0	1	0	0	$\{a, b\}$	a	-1	0	-1
$r_2 = \sqcap_{-1}(r_1, a)$	2	2	1	0	1	0	0	$\{a, b\}$	b	-2	-1	-2
$r_3 = \sqcap_{-2}(r_2, b)$	3	2	1	0	2	0	0	$\{a, b\}$	a	-2	-1	-2
$r_4 = \sqcap_{-2}(r_3, a)$	4	3	2	0	2	0	0	$\{a, b\}$	b	-3	-2	-3
$r_5 = \sqcap_{-3}(r_4, b)$	5	3	2	0	3	0	0	$\{a, b\}$	b	-3	-2	-3
$r_6 = \sqcap_{-3}(r_5, b)$	6	3	2	0	3	0	0	$\{a\}$	a	-3	-1	-2
$r_7 = \sqcap^{-2}(r_6, a)$	6	4	2	0	3	1	0	$\emptyset$				

Table 1: Path in the exploration tree for the generation of the region in Figure 6(b).

Figure 6 presents an example of calculation of minimal regions. Starting from $\mathsf{ER}(a)$, the multiset is iteratively enlarged until a minimal region is obtained. Table 1 describes one of the paths of the exploration tree.

3.1 Symbolic representation of multisets

All the operations required in the algorithm of Figure 5 to manipulate the multisets can be efficiently implemented by using a symbolic representation. A multiset can be modeled as a vector of Boolean functions, where the function at position i describes the characteristic function of the set of states having cardinality i. Hence, multiset operations (union, intersection and complement) can be performed as logic operations (disjunction, conjunction and complement) on Boolean functions. An array of Binary Decision Diagrams (BDDs) [Bry86] is used to represent implicitly a multiset.

4 Synthesis of Petri nets

The synthesis of Petri nets can be based on the generation of minimal regions described by the algorithm in Figure 5.

For the sake of efficiency, different strategies can be sought for synthesis. They can differ in the way the exploration tree is built. One of the possible strategies would consist in defining upper bounds on the capacity of the regions (k_{max})

and on the gradient of the events (w_{max}). The tree can then be explored without surpassing these bounds. For example, one could start with $k_{max} = g_{max} = 1$ to check whether a safe Petri net can be derived. In case the excitation closure does not hold, the bounds could be increased, etc. Splitting labels could be done when the bounds go too high without finding the excitation closure.

By tuning the search algorithm with different parameters, the search for minimal regions can be pruned at the convenience of the user.

Once all minimal regions have been generated, excitation closure must be verified (see Definition 18). In case excitation closure holds, a minimal *saturated*[8] PN can be generated as follows:

- For each event e, a transition labeled with e is generated.
- For each minimal region r_i, a place p_i is generated.
- Place p_i contains k tokens in the initial marking if $r_i(s_{in}) = k$.
- For each event e and minimal region r_i such that $r_i \in {}^\circ e$ find $q \in^\star e$ such that $q = \top_k(r_i)$. Add an arc from place p_i to transition e with weight k. In case $\Delta_{r_i}(e) > -k$ add an arc from transition e to place p_i with weight $k + \Delta_{r_i}(e)$.
- For each event e and minimal region r_i such that $\mathsf{SR}(e) \subseteq r_i$ add an arc from transition e to place p_i with weight $\Delta_{r_i}(e)$[9].

Given that the approach presented in this paper is a generalization for the case of safe Petri nets from [CKLY98], the following theorem can be proved:

Theorem 4. *Let* TS *be an excitation closed transitions system. The synthesis approach of this section derives a* PN *with reachability graph bisimilar to* TS*.*

Proof. In [CKLY98], a proof for the case of safe Petri nets was given (Theorem 3.4). The proof for the bounded case is a simple extension, where only the definition of excitation closure must be adapted to deal with multisets. Due to the lack of space we sketch the steps to attain the generalization.

The proof shows, by induction on the length of the traces leading to a state, that there is a correspondence between the reachability graph of the synthesized net and TS. The crucial idea is that non-minimal regions can be removed from a state in TS to derive a state in the reachability graph of the synthesized net. Moreover excitation closure ensures that this correspondence preserves the transitions in both transition systems. Based on this correspondence, a bisimulation is defined between the states of TS and the states of the reachability graph of the synthesized net. □

The algorithm described in this section is complete in the sense that if excitation closure holds in the initial transition system and a bound large enough is used, the computation of a set of minimal regions to derive a Petri net with bisimilar behavior is guaranteed.

[8] The net synthesized by the algorithm is called saturated since all regions are mapped into the corresponding places [CKLY98].

[9] If $\mathsf{SR}(e) \subseteq r_i$, then the Definition 13 makes $\Delta_{r_i}(e) \geq 0$.

4.1 Irredundant Petri nets

A minimal saturated PN can be redundant. There are two types of redundancies that can be considered in a minimal saturated PN:

1. A minimal region is not necessary to guarantee the excitation closure of any of the events for which the ER is included in it. In this case, the corresponding place can be simply removed from the PN.
2. Given a minimal region r, its corresponding place p and an event e such that $\mathsf{ER}(e) \subseteq \top_k(r)$ and $\Delta_r(e) = g$, if the excitation closure can still be ensured for e by considering $\top_{k-1}(r)$ instead of $\top_k(r)$, then the arcs $p \xrightarrow{k} e$ and $e \xrightarrow{k+g} p$ in the PN can be substituted by the arcs $p \xrightarrow{k-1} e$ and $e \xrightarrow{k+g-1} p$ respectively as long as $k + g - 1 \geq 0$. In the case that $k + g - 1 = 0$, the arc $e \rightarrow p$ can be removed also.

In fact, the first case of redundancy can be considered as a particular case of the second. If we consider that $\top_0(r) = S$, we can say that a region r is redundant when we can ensure the excitation closure of all events by using always $\top_0(r)$ instead of $\top_k(r)$ for some $k > 0$. Note that in this case, the region would be represented by an isolated place (all arcs have weight 0) that could be simply removed from the Petri net without changing its behavior.

5 Splitting events

Splitting events is necessary when the excitation closure does not hold. When an event is split into different copies this corresponds to different transitions in the Petri net with the same label. This section presents some heuristics to split events.

5.1 Splitting disconnected ERs

Assume that we have a situation such as the one depicted in Figure 7 in which

$$EC(e) = \bigcap_{q \in {}^*e} supp(q) \neq \mathsf{ER}(e)$$

However, $\mathsf{ER}(e)$ has several disconnected components $\mathsf{ER}_i(e)$. In the figure, the white part in a $\mathsf{ER}_i(e)$ represents those states in $EC(e) - \mathsf{ER}_i(e)$. For some of those components, $EC(e)$ has no states adjacent to $\mathsf{ER}_i(e)$ ($\mathsf{ER}_2(e)$ in the Figure). By splitting event e into two events e_1 and e_2 in such a way that e_2 corresponds to ERs with no adjacent states in $EC(e)$, we will ensure at least the excitation closure of e_2.

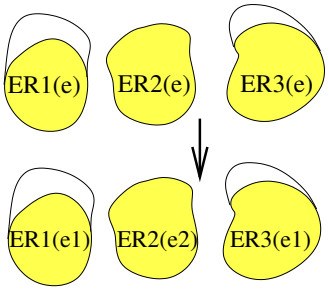


Fig. 7: Disconnected ERs.

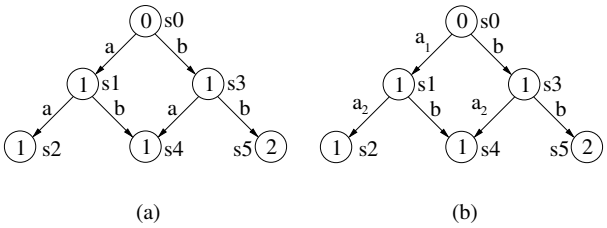


Fig. 8: Splitting on different gradients.

5.2 Splitting on the most promising expansion of an ER

The algorithm presented in Section 3 for generating minimal regions explores all the expansions $\sqcap_k$ and $\sqcap^k$ of an ER. When excitation closure does not hold, all these expansions are stored. Finally, given an event without excitation closure, the expansion r containing the maximum number of events with constant gradient (i.e. the expansion where less effort is needed to transform it into a region by splitting) is selected as source of splitting.

Given the selected expansion r where some events have non-constant gradient, let $|\triangle_r(e)|$ represent the number of different gradients for event e in r. The event a with minimal $|\triangle_r(a)|$ is selected for splitting. Let $g_1, g_2, \ldots g_n$ be the different gradients for a in r. Event a is split into n different events, one for each gradient g_i. Let us use the expansion depicted in Figure 8(a) to illustrate this process. In this multiset, events a and b have both non-constant gradient. The gradients for event a are $\{0, +1\}$ whereas the gradients for b are $\{-1, 0, +1\}$. Therefore event a is selected for splitting. The new events created correspond to the following ERs (shown in Figure 8(b)):

$$\begin{aligned} \mathsf{ER}(a_1) &= \{s_0\} \\ \mathsf{ER}(a_2) &= \{s_1, s_3\} \end{aligned}$$

Intuitively, the splitting of the event with minimal $|\triangle_r(e)|$ represents the minimal necessary legalization of some illegal event in r in order to convert it into a region.

6 Synthesis examples

This section contains some examples of synthesis, illustrating the power of the synthesis algorithm developed in this paper. We show three examples, all them with non-safe behavior, and describe intermidiate solutions that combine splitting with k-bounded synthesis.

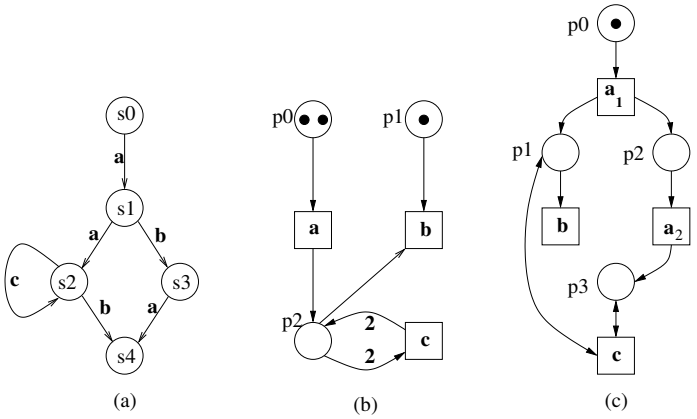


Fig. 9: (a) transition system, (b) 2-bounded Petri net, (c) safe Petri net.

6.1 Example 1

The example is depicted in Figure 9. If 2-bounded synthesis is applied, three minimal regions are sufficient for the excitation closure of the transition system:

$$p_0 = \{s_0^2, s_1, s_3\}, \qquad p_1 = \{s_0, s_1, s_2\}, \qquad p_2 = \{s_1, s_2^2, s_4\}$$

In this example, the excitation closure of event c is guaranteed by the intersection of the multisets $\top_1(p_1)$ and $\top_2(p_2)$, both including $\mathsf{ER}(c)$. However, $\top_1(p_1)$ is redundant and the self-loop arc between p_1 and c can be omitted. The Petri net obtained is shown in Figure 9(b). If safe synthesis is applied, event a must be splitted. The four regions guaranteeing the excitation closure are:

$$p_0 = \{s_0\}, \qquad p_1 = \{s_1, s_2\}, \qquad p_2 = \{s_1, s_3\}, \qquad p_3 = \{s_2, s_4\}$$

The safe Petri net synthesized is shown in Figure 9(c).

6.2 Example 2

The example is shown in Figure 10. In this case, two minimal regions are sufficient to guarantee the excitation closure when the maximal bound allowed is three (the corresponding Petri net is shown in Figure 10(b)):

$$p_0 = \{s_0^3, s_1^2, s_2, s_3\}, \qquad p_1 = \{s_1, s_2^2, s_3, s_4^3, s_5^2, s_6\}$$

The excitation closure of event b is guaranteed by $\top_2(p_1)$. However we also have that $\Delta_b(p_1) = -1$, thus requiring an arc from p_1 to b with weight 2 and another arc from b to p_1 with weight 1. The former is necessary to ensure that b will not fire unless two tokens are held in p_1. The latter is necessary to ensure the gradient -1 of b with regard to p_1. Figures 10(c)-(d) contain the synthesis for bound 2 and 1, respectively.

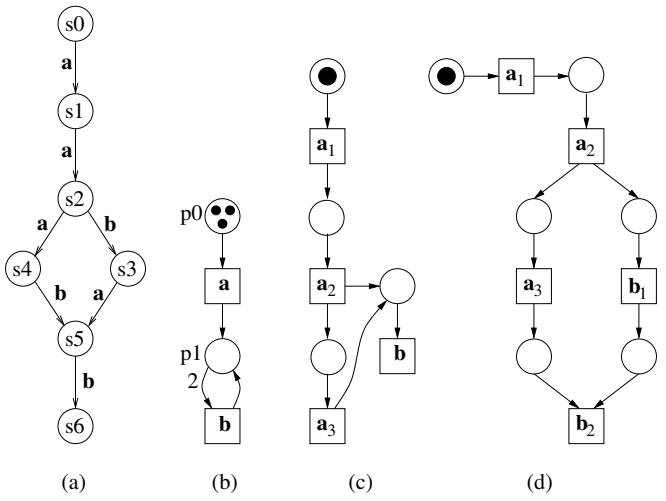


Fig. 10: (a) transition system, (b) 3-bounded, (c) 2-bounded and (d) safe Petri net.

6.3 Example 3

This example is depicted in Figure 11. Using bound 4, the minimal regions are the following:

$$p_0 = \{s_0\}, \qquad p_1 = \{s_1^4, s_2^3, s_3, s_4^2, s_6\}, \qquad p_2 = \{s_2, s_4^2, s_5, s_6^3, s_7^4\}$$

The synthesis with different bounds is shown in Figures 11(c)-(d). Notice that the synthesis for bounds two and three derives the same Petri net.

7 Experimental results

In this section a set of parameterizable benchmarks are synthesized using the methods described in this paper. The following examples have been artificially created:

1. A model for n processes competing for m shared resources, where $n > m$. Figure 12(a) describes a Petri net for this model[10],
2. A model for m producers and n consumers, where $m > n$. Figure 12(b) describes a Petri net for this model.
3. A 2-bounded pipeline of n processes. Figure 12(c) describes a Petri net for this model.

Table 2 contains a comparison between a synthesis algorithm of safe Petri nets [CKLY98], implemented in the tool `petrify`, and the synthesis of general

[10] A simplified version of this model was also synthesized by SYNET [Cai02] in [BD98].

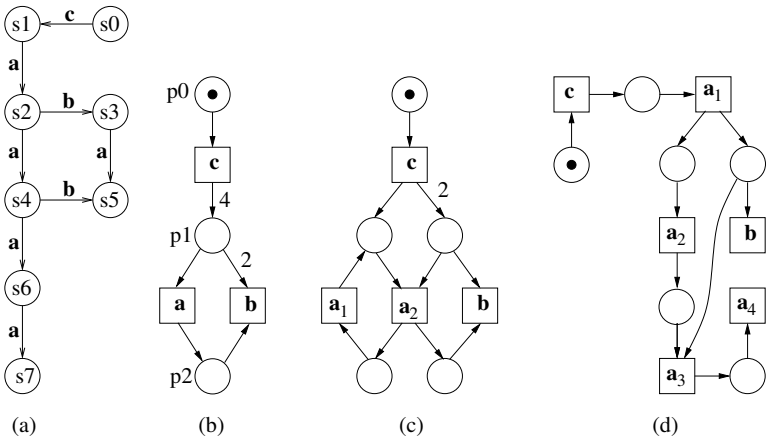


Fig. 11: (a) transition system, (b) 4-bounded, (c) 3-bounded/2-bounded and (d) safe Petri net.

Petri nets as described in this paper, implemented in the prototype tool `Genet`. For each benchmark, the size of the transition system (states and arcs), number of places and transitions and cpu time is shown for the two approaches. The transition system has been initially generated from the Petri nets. Clearly, the methods developed in this paper generalize those of the tool `petrify`, and particularly the generation of minimal regions for arbitrary bounds has significantly more complexity than its safe counterpart. However, many of the implementation heuristics and optimizations included in `petrify` must also be extended and adapted to `Genet`. Provided that this optimization stage is under development in `Genet`, we concentrate on the synthesis of small examples. Hence, cpu times are only preliminary and may be improved after the optimization of the tool.

The main message from Table 2 is the expressive power of the approach developed in this paper to derive an event-based representation of a state-based one, with minimal size. If the initial transition system is excitation closed, using a bound large enough one can guarantee no splitting and therefore the number of events in the synthesized Petri net is equal to the number of different events in the transition system. Note that the excitation closure holds for all the benchmarks considered in Table 2, because the transition systems considered are derived from the corresponding Petri nets shown in Figure 12.

Label splitting is a key method to ensure the excitation closure. However, its application can degrade the solution significantly (both in terms of cpu time required for the synthesis and the quality of the solution obtained), specially if many splittings must be performed to achieve excitation closure, as it can be seen in the benchmarks SharedResource(5,2) and BoundedPipeline(7). In these examples, the synthesis performed by `petrify` derives a safe Petri net with one order of magnitude more transitions than the bounded synthesis method presented in this paper. Hence label splitting might be relegated to situations

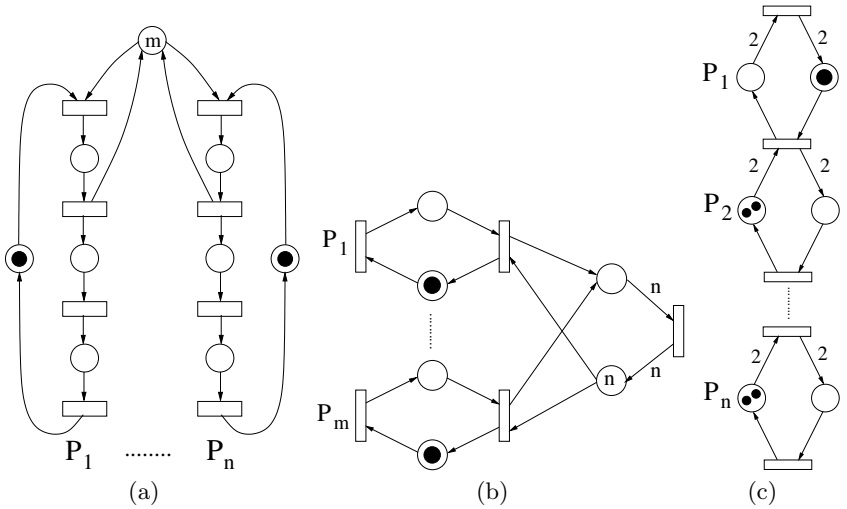


Fig. 12: Parameterized benchmarks: (a) n processes competing for m shared resources, (b) m producers and n consumers, (c) a 2-bounded pipeline of n processes.

where the excitation closure does not hold (see the examples of Section 6), or when the maximal bound is not known, or when constraints are imposed on the bound of the resulting Petri net.

8 Conclusions

An algorithm for the synthesis of k-bounded Petri nets has been presented. By heuristically splitting events into multiple transitions, the algorithm always guarantees a visualization object. Still, a minimal Petri net with bisimilar behavior is obtained when the original transition systems is excitation closed.

The theory presented in this paper is accompanied with a tool that provides a practical visualization engine for concurrent behaviors.

References

[BBD95] E. Badouel, L. Bernardinello, and P. Darondeau. Polynomial algorithms for the synthesis of bounded nets. *Lecture Notes in Computer Science*, 915:364–383, 1995.

[BD98] Eric Badouel and Philippe Darondeau. Theory of regions. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 529–586. Springer, 1998.

[BDLS07] R. Bergenthum, J. Desel, R. Lorenz, and S.Mauser. Process mining based on regions of languages. In *Proc. 5th Int. Conf. on Business Process Management*, pages 375–383, September 2007.

[Bry86] Randal Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computer-Aided Design*, 35(8):677–691, 1986.

			Petrify			Genet		
benchmark	$\|S\|$	$\|E\|$	$\|P\|$	$\|T\|$	CPU	$\|P\|$	$\|T\|$	CPU
SharedResource(3,2)	63	186	15	16	0s	13	12	0s
SharedResource(4,2)	243	936	20	24	5s	17	16	21s
SharedResource(5,2)	918	4320	48	197	180m	24	20	6m
SharedResource(4,3)	255	1016	21	26	2s	17	16	4m40s
ProducerConsumer(3,2)	24	68	9	10	0s	8	7	0s
ProducerConsumer(4,2)	48	176	11	13	0s	10	9	0s
ProducerConsumer(3,3)	32	92	10	13	0s	8	7	5s
ProducerConsumer(4,3)	64	240	12	17	1s	10	9	37s
ProducerConsumer(6,3)	256	1408	16	25	25s	14	13	16m
BoundedPipeline(4)	81	135	14	9	0s	8	5	1s
BoundedPipeline(5)	243	459	17	11	1s	10	6	19s
BoundedPipeline(6)	729	1539	27	19	6s	12	7	5m
BoundedPipeline(7)	2187	5103	83	68	110m	14	8	88m

Table 2: Synthesis of parameterized benchmarks

[Cai02] Benoît Caillaud. Synet : A synthesizer of distributable bounded Petri-nets from finite automata. http://www.irisa.fr/s4/tools/synet/, 2002.

[CGP00] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, 2000.

[CKLY98] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri nets from finite transition systems. *IEEE Transactions on Computers*, 47(8):859–882, August 1998.

[Dar07] Philippe Darondeau. Synthesis and control of asynchronous and distributed systems. In Twan Basten, Gabriel Juhás, and Sandeep K. Shukla, editors, *ACSD*, pages 13–22. IEEE Computer Society, 2007.

[DR96] J. Desel and W. Reisig. The synthesis problem of Petri nets. *Acta Informatica*, 33(4):297–315, 1996.

[ER90] A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures. Part I, II. *Acta Informatica*, 27:315–368, 1990.

[HKT95] P. W. Hoogers, H. C. M. Kleijn, and P. S. Thiagarajan. A trace semantics for petri nets. *Inf. Comput.*, 117(1):98–114, 1995.

[HKT96] P. W. Hoogers, H. C. M. Kleijn, and P. S. Thiagarajan. An event structure semantics for general petri nets. *Theor. Comput. Sci.*, 153(1&2):129–170, 1996.

[Maz87] Antoni W. Mazurkiewicz. Trace theory. In *Advances in Petri Nets*, volume 255 of *Lecture Notes in Computer Science*, pages 279–324, 1987.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Muk92] M. Mukund. Petri nets and step transition systems. *Int. Journal of Foundations of Computer Science*, 3(4):443–478, 1992.

[SBY07] D. Sokolov, A. Bystrov, and A. Yakovlev. Direct mapping of low-latency asynchronous controllers from STGs. *IEEE Transactions on Computer-Aided Design*, 26(6):993–1009, June 2007.

[VPWJ07] H.M.W. Verbeek, A.J. Pretorius, W.M.P. van der Aalst, and J.J. van Wijk. On Petri-net synthesis and attribute-based visualization. In *Proc. Workshop on Petri Nets and Software Engineering (PNSE'07)*, pages 127–141, June 2007.