Decomposition of transition systems into sets of synchronizing Free-choice Petri Nets

Viktor Teren
Department of Computer Science
Università degli Studi di Verona
Verona, Italy
viktor.teren@univr.it

Jordi Cortadella
Department of Computer Science
Universitat Politècnica de Catalunya
Barcelona, Spain
jordi.cortadella@upc.edu

Tiziano Villa
Department of Computer Science
Università degli Studi di Verona
Verona, Italy
tiziano.villa@univr.it

***Abstract*—Petri nets and transition systems are two important formalisms used for modeling concurrent systems. One interesting problem in this domain is the creation of a Petri net with a reachability graph equivalent to a given transition system. This paper focuses on the creation of a set of synchronizing Free-choice Petri nets (FCPNs) from a transition system. FCPNs are more amenable for visualization and structural analysis while not being excessively simple, as in the case of state machines. The results show that with a small set of FCPNs, the complexity of the model can be reduced when compared to the synthesis of a monolithic Petri net.**

***Index Terms*—Transition system, Petri net, Free-choice Petri net, decomposition, theory of regions, SAT, pseudo-Boolean optimization.**

I. Introduction

The problem of synthesizing a Petri net (PN) from a transition system (TS) has been studied in the past [1], [2]. The main goal of the proposed techniques [3] is to derive a PN with a minimum place/transition count. This goal often results in the synthesis of tangled nets with a non-intuitive behavior that is difficult to visualize.

Free-choice Petri nets (FCPNs) are a subclass of Petri nets exhibiting structural properties that make their visualization and analysis simpler. Visualization is simpler because FCPNs can capture causality, choice and concurrency: elementary structures which avoid complex intertwined PN representations. At the same time, concurrency allows representing more complex sequential behaviors than those of state machines. Analysis is simpler, instead, because several important problems can be solved in polynomial time [4, p. 403]. However, not every behavior can be represented as an FCPN.

In this paper, we propose a method to decompose a behavior into a set of FCPNs such that their synchronization reproduces the original behavior. The goal is to avoid the analysis of complex monolithic PNs and focus on the analysis of a small set of simpler PNs that can be easily visualized.

The theory of regions [5] has been used for this synthesis problem. After creating a set of regions (places) that can be used for a monolithic PN, they are split into non-disjoint sets, in such a way that each set of regions produces an FCPN. The FCPNs synchronize via the common events. The goal of this strategy is twofold: (1) produce a small number of nets and (2) reduce the complexity of each net. Section III shows how these goals can be achieved.

A. Previous vs current work

In [6], we showed how to decompose into sets of synchronizing state machines (SMs). Both strategies are based on the theory of regions and use the same method for the extraction of regions from the TS. The fundamental difference resides on how the regions are combined to derive a set of synchronizing PNs. In [6], a set of SMs is produced: the main limitation of the restriction to SMs is their sequential nature (SMs model only causality and choice), i.e., do not capture concurrency, and concurrent events must be split in different components. This paper mitigates such restriction by allowing concurrent events to be in the same component, under the constraint that the component must be free-choice. With the new approach, the number of components is reduced, trading-off number vs. complexity of components. The step from SMs to FCPNs requires the introduction of a more complex set of constraints on the decomposition process, as we will explain in detail in Sec. III, especially in Sec. III-C.

B. Overview

We give an overview of the method proposed in this paper and illustrate the main contribution using a simple example.

Figure 1 depicts a TS and a PN net with an equivalent behavior. A concurrent system often represents the cooperation of different subsystems that interact through common events. It is interesting to identify and distill the components of the

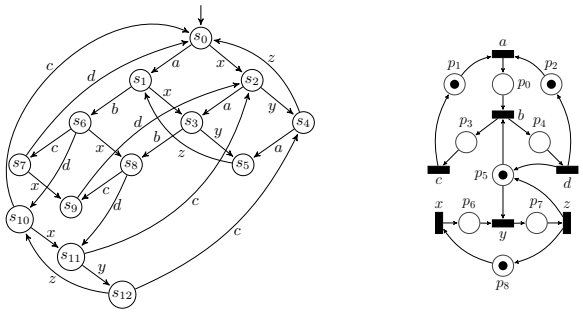
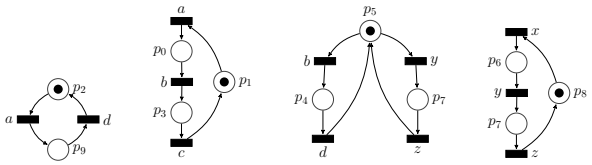


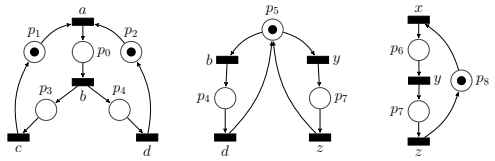
Fig. 1: Transition system and Petri net.

 DOI 10.1109/DSD57027.2022.00031



(a) SM_0 (b) SM_1 (c) SM_2 (d) SM_3

Fig. 2: Four SMs distilled from the TS in Fig. 1



(a) $FCPN_0$ (b) $FCPN_1$ (c) $FCPN_2$

Fig. 3: Three FCPNs distilled from the TS in Fig. 1

system in a way that they can be visualized and analyzed individually, thus hiding the other components.

This paper is an attempt to perform a distillation without prior knowledge of the components, with two goals: (1) extract subsystems with nice structural properties, i.e., easy to visualize and analyze and (2) guarantee that their composition reproduces the original behavior of the system.

Figure 2 depicts an SM-decomposition as proposed in [6]. Due to the inherent concurrency of the system, the decomposition requires four SMs to fully capture the behavior.

In this paper, we allow incorporating concurrency into the components by extending their structural properties. Figure 3 shows three components extracted from the TS in Fig. 1. It can be observed that the system models the interaction of two components with alphabets $\{a, b, c, d\}$ and $\{x, y, z\}$, respectively, shown at the left (a) and right (c) of the picture. In the middle, there is an arbitration process (b) that creates a mutual exclusion between $\{b, d\}$ and $\{y, z\}$.

This example illustrates the purpose of this work: extract hidden knowledge from complex systems. In this case, we enforce the components to have structural properties that are formally captured by the concept of FCPN, which combines causality, concurrency and choice with simple structural rules.

The example also illustrates the main contribution of this work with regard to the SM-decomposition proposed in [6]. The use of FCPNs allows reducing the number of components while preserving nice structural properties.

The possibility of extracting hidden knowledge from unstructured data suggests that our decomposition may be applied to process mining. A TS can be created directly from a set of traces [7], [8] and then given as input to our decomposition algorithm. As mentioned previously, there are advantages in decomposing into a set of FCPNs, rather than representing the entire behavior with only one PN, especially when the PN is very complex. The application of TS decomposition to process mining as an integration to established process mining techniques is ongoing research to be reported in the future.

This paper is organized as follows. Sec. II introduces the background theory, with a special reference to theory of regions. Sec. III describes the proposed decomposition flow, specifying the constraints used in the intermediate steps. Sec. IV argues the equivalence by bisimulation of the composition of FCPNs to the original TS (proof in the appendix). Sec. V reports our experimental results; conclusions follow in Sec. VI.

II. Preliminaries

A. Transition systems

Definition 1 (TS/LTS)**.** [2] A Labeled Transition System (LTS, or simply TS) is defined as a 4-tuple (S, E, T, s_0) where:

- S is a non-empty set of states
- E is a set of events/labels
- $T \subseteq S \times E \times S$ is a transition relation
- $s_0 \in S$ is an initial state

Every TS is supposed to satisfy the following properties:

- It does not contain-self loops: $\forall (s, e, s') \in T : s \neq s'$;
- Each event has at least one occurrence: $\forall e \in E : \exists (s, e, s') \in T$;
- Every state is reachable from the initial state: $\forall s \in S : s_0 \rightarrow^* s$, i.e. $\exists (s_0, e_0, s_1), (s_1, e_1, s_2), \ldots, (s_n, e_n, s) \in T$;
- It is deterministic: for each state there is at most one successor state reachable with label e.

Fig. 1 shows the transition system used as a running example to illustrate the decomposition procedure.

Definition 2 (Bisimulation)**.** Given two transition systems $TS_1 = (S_1, E, T_1, s_{0,1})$ and $TS_2 = (S_2, E, T_2, s_{0,2})$, a binary relation $B \subseteq S_1 \times S_2$ is a bisimulation, denoted by $TS_1 \sim_B TS_2$, if $(s_{0,1}, s_{0,2}) \in B$ and if whenever $(p, q) \in B$:

- $\forall (p, e, p') \in T_1 : \exists q' \in S_2$ such that $(q, e, q') \in T_2$ and $(p', q') \in B$
- $\forall (q, e, q') \in T_2 : \exists p' \in S_1$ such that $(p, e, p') \in T_1$ and $(p', q') \in B$.

Two TSs are said to be bisimilar if there is a bisimulation between them.

The operation *Reach* deletes from a TS all the states that are not reachable or accessible from the initial state and all transitions attached to them.

Definition 3 (Synchronous product)**.** Given two TSs, $TS_1 = (S_1, E_1, T_1, s_{0,1})$ and $TS_2 = (S_2, E_2, T_2, s_{0,2})$, the synchronous product is defined as

$$TS_1 || TS_2 = \mathit{Reach}(S, E_1 \cup E_2, T, (s_{0,1}, s_{0,2})),$$

where $S = S_1 \times S_2$ and $(s_{0,1}, s_{0,2}) \in S$.
$T \subseteq (S_1 \times S_2) \times E \times (S_1 \times S_2)$ is defined as follows:

- if $a \in E_1 \cap E_2$, $(s_1, a, s_1') \in T_1$, and $(s_2, a, s_2') \in T_2$, then $((s_1, s_2), a, (s_1', s_2')) \in T$,
- if $a \in E_1$, $a \notin E_2$, and $(s_1, a, s_1') \in T_1$, then $((s_1, s_2), a, (s_1', s_2)) \in T$,
- if $a \notin E_1$, $a \in E_2$, and $(s_2, a, s_2') \in T_2$, then $((s_1, s_2), a, (s_1, s_2')) \in T$,

- nothing else belongs to T.

The *Reach* operator is required for the synchronous product: even if all states of TS_1 and TS_2 are reachable, some states derived from the Cartesian product could be unreachable from the initial state, given the definition of T.

The synchronous product is associative, so we can define the product of a collection of n TSs: $TS_1||TS_2||\dots||TS_n = ((TS_1||TS_2)\dots)||TS_n$.

B. Petri Nets

We assume the reader to be familiar with Petri nets. We refer to [9] for a deeper insight on the concepts used in this work. This section introduces the nomenclature related to Petri nets used along the paper.

In this work we will only deal with safe Petri nets, i.e., nets whose places do not contain more than one token in any reachable marking. For this reason, we will model markings as sets of places.

Definition 4 (Ordinary Petri Net)**.** [9] An ordinary Petri net is a 4-tuple, $PN = (P, T, F, M_0)$ where:

- $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places,
- $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions,
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),
- $M_0 \subseteq P$ is an initial marking,
- $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

For any $x \in P \cup T$, then ${}^\bullet x = \{y | (y, x) \in F\}$. Similarly, $x^\bullet = \{y \mid (x, y) \in F\}$.

Definition 5 (Firing rule)**.** [1, p. 17] Let $N = (P, T, F, M_0)$ be a safe Petri net. A transition $t \in T$ enabled in marking M is represented as $M[t\rangle$. If t is enabled in M, then t can be fired leading to another marking M', denoted as $M[t\rangle M'$, such that $M' = M \backslash {}^\bullet t \cup t^\bullet$.

We call $[M\rangle$ the set of markings that can be reached from M by firing sequences of enabled transitions.

Definition 6 (Reachability graph)**.** [1, p. 20] Given a safe Petri net $N = (P, T, F, M_0)$, the reachability graph of N is the transition system $RG(N) = ([M_0\rangle, T, \Delta, M_0)$ defined by $(M, t, M') \in \Delta$ if $M \in [M_0\rangle$ and $M[t\rangle M'$.

Definition 7 (Free-choice Petri net)**.** [9] A Free-choice Petri net is an ordinary Petri net $N = (P, T, F, M_0)$ such that every arc from a place is either a unique outgoing arc or a unique incoming arc to a transition, i.e.,

for all $p \in P$, $|p^\bullet| \leq 1$ or ${}^\bullet(p^\bullet) = \{p\}$; equivalently,
for all $p_1, p_2 \in P$, $p_1^\bullet \cap p_2^\bullet \neq \emptyset \Rightarrow |p_1^\bullet| = |p_2^\bullet| = 1$.

In this paper, we consider sets of synchronized FCPNs.

C. From LTS to Petri nets by regions

In this paper we propose a procedure for the decomposition of Transition Systems based on the theory of regions (from [2]). A region is a subset of states in which all the transitions under the same event have the same relation with the region: either all entering, or all exiting, or not crossing the region.

Definition 8 (Region)**.** Given a $TS = (S, E, T, s_0)$, a region is defined as a non-empty set of states $r \subsetneq S$ such that the following properties hold for each event $e \in E$:

$$\begin{aligned} enter(e,r) &\implies \neg in(e,r) \wedge \neg out(e,r) \wedge \neg exit(e,r) \\ exit(e,r) &\implies \neg in(e,r) \wedge \neg out(e,r) \wedge \neg enter(e,r) \end{aligned}$$

where:

$$\left.\begin{aligned} in(e,r) &\equiv \exists (s,e,s') \in T : s, s' \in r \\ out(e,r) &\equiv \exists (s,e,s') \in T : s, s' \notin r \end{aligned}\right\} \textit{no\_cross}$$
$$\begin{aligned} enter(e,r) &\equiv \exists (s,e,s') \in T : s \notin r \wedge s' \in r \\ exit(e,r) &\equiv \exists (s,e,s') \in T : s \in r \wedge s' \notin r \end{aligned}$$

Definition 9 (Minimal region)**.** A region r is *minimal* if there is no other region r' strictly contained in r ($\nexists r' \mid r' \subset r$).

The minimal regions of the TS in Fig. 1 are shown in Table I.

Region	States of the TS
r_0	$\{s_1, s_3, s_5\}$
r_1	$\{s_0, s_2, s_4, s_7, s_9\}$
r_2	$\{s_0, s_2, s_4, s_{10}, s_{11}, s_{12}\}$
r_3	$\{s_6, s_8, s_{10}, s_{11}, s_{12}\}$
r_4	$\{s_6, s_7, s_8, s_9\}$
r_5	$\{s_0, s_1, s_2, s_3, s_{10}, s_{11}\}$
r_6	$\{s_2, s_3, s_8, s_9, s_{11}\}$
r_7	$\{s_4, s_5, s_{12}\}$
r_8	$\{s_0, s_1, s_6, s_7, s_{10}\}$

TABLE I: Minimal regions of the TS in Fig. 1

Theorem 1. If r_1 and r_2 are non-disjoint regions then $r_1 \cup r_2$ is a region iff $r_1 \cap r_2$ is a region.

Proof. See the appendix. □

Definition 10 (Pre-region (Post-region))**.** A region r is a pre-region (post-region) of an event e if there is a transition labeled with e which exits r (enters r). The set of all pre-regions (post-regions) of the event e is denoted by ${}^\circ e$ (e°).

By definition, if $r \in {}^\circ e$ ($r \in e^\circ$) all the transitions labeled with e are exiting r (entering r). Furthermore, if the transition system is strongly connected, all the regions are also pre-regions of some event.

Definition 11 (Excitation set / Switching set)**.** The *excitation (switching) set* of event e, $ES(e)$ ($SS(e)$), is the maximal set of states such that for every $s \in ES(e)$ ($s \in SS(e)$) there is a transition $s \xrightarrow{e}$ ($\xrightarrow{e} s$).

The excitation sets of the TS in Fig. 1 are reported in Table II.

Event	Pre-regions	ES(event)
a	$\{r_1, r_2\}$	$\{s_0, s_2, s_4\}$
b	$\{r_0, r_5\}$	$\{s_1, s_3\}$
c	$\{r_3\}$	$\{s_{10}, s_{11}, s_{12}\}$
d	$\{r_4\}$	$\{s_6, s_7, s_8, s_9\}$
x	$\{r_8\}$	$\{s_0, s_1, s_6, s_7, s_{10}\}$
y	$\{r_5, r_6\}$	$\{s_2, s_3, s_{11}\}$
z	$\{r_7\}$	$\{s_4, s_5, , s_{12}\}$

TABLE II: Pre-regions and excitation sets for the TS in Fig. 1

Definition 12 (Excitation-closed Transition System (ECTS))**.** A TS with the set of labels E and the pre-regions ${}^\circ e$ is an ECTS if the following conditions are satisfied:

- Excitation closure: $\forall e \in E : \bigcap_{r \in {}^\circ e} r = ES(e)$
- Event effectiveness: $\forall e \in E : {}^\circ e \neq \emptyset$

The excitation set is very important because it represents the starting point for the creation of a region: a region always corresponds to the superset of the excitation set of one or more events. Furthermore, the excitation set is used to check if a TS satisfies the excitation closure. The EC property guarantees that two states, s_i and s_j, are bisimilar by checking that they cannot be *separated* by any region, i.e., there is no minimal region r such that $s_i \in r$ and $s_j \notin r$.

If the initial TS does not satisfy the excitation closure (EC) or event effectiveness property, *label splitting* [2] can be performed to obtain an ECTS. Label splitting means splitting a label which does not satisfy excitation closure into two different labels trying to solve some of the violated constraints. Label splitting is performed until excitation closure is achieved i.e. there are no more events violating excitation closure.

In our example, there is no need for label splitting. It is easy to check that the EC holds for all events of the TS, e.g., $ES(a) = r_1 \cap r_2$, and similarly for the rest of events.

The synthesis of a Petri net from an ECTS, proposed in [2], can be summarized by the following steps:

1) *Generation of all minimal regions*. The excitation sets are expanded until they become regions.
2) *Removal of redundant regions*. Some minimal regions may be redundant, meaning that they can be removed while the EC property still holds.
3) *Merging minimal regions*. Disjoint minimal regions are merged into non-minimal regions, thus reducing the number of places. This merging can be done as long as the EC is preserved.
4) *PN creation*. Regions and events are converted into places and transitions. A token is assigned to places derived from regions containing the TS's initial state. The arcs between places and transitions are added following the *enter* and *exit* properties of the related regions and events.

III. Algorithm for the generation of FCPNs

Here first we define what is an excitation-closed set of Free-choice Petri nets derived from an ECTS, and next we will see how to generate the aforementioned decomposition.

Definition 13 (Excitation-closed set of Free-choice Petri nets derived from an ECTS)**.** Given a set of FCPNs N derived from an ECTS *TS*, the set of all regions R of N, the set of labels E of *TS*, the sets of pre-regions ${}^\circ e$ of the *TS* for all $e \in E$:

N is excitation-closed with respect to *TS* if the following conditions are satisfied:

- Excitation closure: $\forall e \in E : \bigcap_{r \in ({}^\circ e \cap R)} r = ES(e)$
- Event effectiveness: $\forall e \in E : \exists r \in R \mid r \in {}^\circ e$

Input: *TS*: a transition system
Output: N: a set of FCPNs
$R \leftarrow$ GenerateRegions(TS) ; `// Set of regions`
$N \leftarrow \emptyset$; `// Set of FCPNs`
$R_N \leftarrow \emptyset$; `// Regions in the extracted FCPNs`
repeat
  $\widehat{R} \leftarrow$ ExtractFCPN(*TS*, R, R_N);
  $N \leftarrow N \cup \{$PN(*TS*, $\widehat{R})\}$;
  $R_N \leftarrow R_N \cup \widehat{R}$;
until EC(*TS*, R_N);
PetriNetSimplification(N);

Algorithm 1: Main algorithm for the generation of FCPNs.

Algorithm 1 sketches the methods to derive a set of FCPNs from a TS.

Initially, the set of minimal regions R is generated. This set can be directly used to derive a monolithic PN [3]. Instead, we want to generate a set of synchronizing FCPNs. The FCPNs are extracted sequentially, one after the other, until the set of regions (places) used in the FCPNs R_N fulfills the excitation closure respect to the initial transition system *TS*, i.e., EC(*TS*, R_N).

It is important to realize that not all minimal regions are always required to satisfy the EC, i.e., some regions in R might be not taken after all runs of ExtractFCPN(*TS*, R, R_N).

The core of the algorithm is the function ExtractFCPN, that generates a new FCPN considering the set of regions R and the set of regions, R_N, that have already been used in previous FCPNs. The function PN(*TS*, $\widehat{R}$) generates a PN from the set of regions $\widehat{R}$ returned by ExtractFCPN. The PN is generated by including all those events that enter/exit the regions in $\widehat{R}$[1].

A final simplification step is included to reduce the complexity of the FCPNs.

A. Extraction of one FCPN

The extraction of one FCPN consists of identifying a new set of places, $\widehat{P}$, such that two conditions are met:

- PN(TS, $\widehat{P}$) is an FCPN.
- $|\widehat{P} \setminus P_N|$ is maximized.

The extraction of one FCPN can be modeled as a Pseudo-Boolean optimization problem [10]. Each region (place) $r \in R$ and each event $e \in E$ are represented by one Boolean variable. When r is asserted, it means that $r \in \widehat{P}$. The constraints to derive an FCPN are the following[2]:

- *Place connectedness*. If a region $r \in \widehat{P}$, then all incoming and outgoing events are also present in the FCPN.
- *Event connectedness*. If an event e is present, at least one pre-region and one post-region of e must belong to $\widehat{P}$.
- *Free-choiceness*. if r is a choice present in the FCPN, then r must be the only selected pre-region of its post-events. Formally, if $r \in \widehat{P}$, $|r^\bullet| > 1$, $e \in r^\bullet$, $r' \in {}^\bullet e$, and $r \neq r'$, then $r' \notin \widehat{P}$.

[1]The details of this function are not shown. We assume the reader can easily figure out what is the PN induced by a subset of regions (places).

[2]Each constraint is modeled by a Boolean formula represented as a set of clauses in CNF form.

These FCPN constraints are modeled as constraints between regions and their enter/exit events, in particular *Place connectedness* is necessary for the correct representation by places of the connections between the regions which they represent and their enter/exit events. Also both *Place connectedness* and *Event connectedness* are necessary for the creation of *bounded* PNs. Removing the latter constraints, we might have a decomposition into a set of unbounded FCPNs, but still with a bounded synchronous composition.

The maximization of $|\widehat{P} \setminus P_N|$ can be modeled by Pseudo-Boolean optimization. It increases the satisfaction of the excitation closure, since the algorithm greedily incorporates a higher number of new places into P_N.

This approach is still heuristic and does not exploit the newly used regions, indeed in rare cases it can fail, as with the example *intel_edge* in Table IV, which achieved a result with four FCPNs, whereas the same heuristic with a different initial choice was able to find a solution with three FCPNs.

Looking into the implementation of all previous constraints, a SAT solver was used. Each of the previously described properties was encoded into a set of CNF clauses, then by incremental search instances of SAT problems were solved in order to find the result of the maximization function (see [10]).

In some particular cases the SAT solver could find a set of places which are not connected i.e. a set of places representing two or more FCPNs without any common place and transition. In this case, the smaller FCPNs are not selected for the current computation in search of the optimal result, whereas the same FCPNs could be individually taken in future iterations.

B. Minimization of the number of FCPNs

Once a set of synchronizing FCPNs is obtained, we can improve our results minimizing the number and the size of the obtained FCPNs; indeed, the next step is the greedy algorithm performed as in the case of State Machines: starting from the largest FCPN, we try to remove it checking the preservation of excitation closure, if excitation closure is preserved then the FCPN can be definitely removed, since the remaining set of FCPNs is sufficient to model the behavior of the TS.

The minimization of the number of FCPNs should reduce the occurrences of the synchronizing events, reducing the redundancy among the set of FCPNs. Furthermore, minimizing the number of FCPNs, excessive fragmentation can be avoided, a problem which in some highly concurrent cases may persist in SM decomposition.

C. Minimization of the size of single FCPNs: merge algorithm

The last minimization step i.e. merge of regions consists of removing events by merging adjacent regions. This procedure further removes the redundant events and places, leaving only events not found in other FCPNs and the events essential for the synchronization between FCPNs. This minimization allows to focus on the uniqueness of each FCPN. An example of a region derived by merge can be seen in Fig. 2(a): p_9 is the result of the removal of event b in SM_0, merging the regions r_0 and r_4 respectively representing places p_0 and p_4. Merge within FCPNs is slight different from merge within SMs: a set of constraints must be added in order to maintain the correct behavior and FCPN property. Here the SAT constraints used for the merge of regions are reported:

1) Preservation of **excitation closure**: all occurrences of each used region can be removed except one occurrence (preserves excitation closure).
2) Effective merge: when removing a label, remove the connected regions.
3) Structural constraint: the merging regions have to be **disjoint** or the intersection of the merging regions has to be a region.
4) Free-choice structure preserving constraint: given a couple of merging regions r_1 and r_2 where event e is removed and r_1 is a pre-region for e and r_2 is a post-region for e, if r_1 is a pre-region for an event e' where $e \neq e'$ and r_2 is a pre-region for an event e'' where $e'' \neq e$ and there exists a pre-region of e'' called r_3 i.e. $r_3 \neq r_2$ then the merge between r_1 and r_2 cannot be done in order to preserve the free-choice structure.
5) Minimization function: number of events.

The constraints 1, 2 and 5 are inherited from SMs.

Constraint 1 is our main constraint. The presence of at least one occurrence of each region guarantees the excitation closure, a sufficient condition in order to achieve a model bisimilar to the initial transition system.

Constraint 2 is a structural constraint created in order to impose the effective removal of labels merging adjacent regions.

Constraint 3 is implicit in case of SMs because an SM by construction is composed of a set of disjoint regions. In case of FCPNs we could have non-disjoint regions, therefore this kind of constraint has to be enforced explicitly in order to be sure that merging regions will still create a region: the following theorem gives a sufficient condition for this to happen, improving the performance during the creation of constraints.

This result justifies the fact that even though the union of two non-disjoint minimal regions may not yield a region, after some iterations which generate non-minimal regions, it may be possible to obtain regions by merging non-disjoint regions.

Constraint 4 is fundamental in order to maintain the free-choice property: merging a couple of regions the PN structure is modified and could contain forbidden structures for the chosen subclass of PNs e. g. a free-choice Petri net which becomes an asymmetric-choice PN (Fig. 4).

The example in Fig. 4 is the simplest case of violation of the free-choice structure; it can be extended to cases with the same structure but more places/transitions, and it represents the only way to lose the free-choice property by merge.

Theorem 2. Minimizing an FCPN, the free-choice property can be lost only by merging two places involving a choice and a join structure.

Proof. Given an already existing structure, to create an asymmetric choice, a choice and a join must be overlapped by

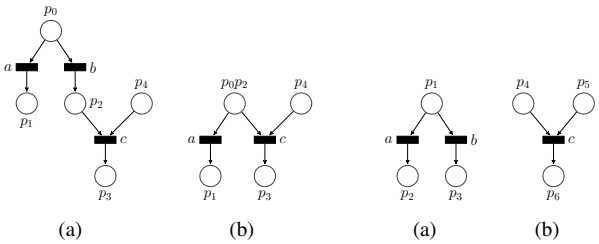


Fig. 4: Example of the loss of free-choice property after the removal of event b.

Fig. 5: Choice and join used for the proof.

a merge. Initially the choice and the join structures cannot have in common a transition because it would mean that we already have an asymmetric choice. If the two structures do not have anything in common the merge on one of them does not affect the structure of the other, therefore in this case the merge is safe. But what happens if the two structures have one or more common places? Considering only the simplest structures we cannot have three common places (overlapping the two structures we can have at most two common places). We enumerate the possible cases when merging a choice with places p_1, p_2 and p_3 and a join with places p_4, p_5 and p_6 (Fig. 5).

Cases with one common place:

1) $p_1 = p_6$: each merge is safe;
2) $p_1 = p_4$ or $p_1 = p_5$: this case is not possible because it already represents an asymmetric choice;
3) $p_6 = p_2$ or $p_6 = p_3$: each possible merge on a, b or c is safe;
4) $p_2 = p_4$ or $p_2 = p_5$ or $p_3 = p_4$ or $p_3 = p_5$: merge on c is safe and the same for the merge on either a or b according to which one is not connected neither to p_4 nor to p_5, otherwise, if either a or b is connected to either p_4 or p_5 we have the forbidden case represented in Fig. 4, e.g, when $p_2 = p_4$ merging on a is forbidden.

Cases with two common places:

5) ($p_1 = p_4$ or $p_1 = p_5$) and ($p_2 = p_6$ or $p_3 = p_6$): this case is not possible because it already represents an asymmetric choice;
6) $p_2 = p_4$ and $p_3 = p_5$ (or $p_2 = p_5$ and $p_3 = p_4$): this is a special case of 4, therefore only event c can be removed, since both events a and b are connected to places of the join construct. This case represents also a deadlock situation therefore it should not occur as input to the merge algorithm. □

Constraint 5 is fundamental in order to find the smallest solution without violating the other constraints; minimization is again performed with binary search calls to a SAT solver.

When the SAT computation is finished not all merges are directly performed, so an additional check is done in order to avoid the unexpected removal of events. Suppose that the SAT solver got as a result a sequence of merges where each merge is valid, but the final result is the removal of an entire choice branch: if the choice on the other side does not contain any

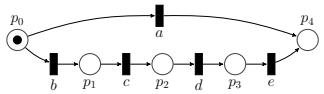


Fig. 6: Removing events b, c, d and e as a collateral effect also event a is removed.

place but only one or more events, then these events would be removed merging the entire set of places (Fig. 6). In this case given a sequence of merging regions one merge is not performed in order to keep both branches.

IV. Composition of FCPNs and equivalence to original TS

In the previous section we showed how to decompose a transition system into a set of synchronizing FCPNs, and we proved that the free-choice structure is preserved until the end of the decomposition process. In this section we prove a structural equivalence between the original TS and the synchronous composition of the reachability graphs of the FCPNs, by defining a bisimulation relation between them, under the assumptions of excitation-closure and place-connectedness of the FCPNs.

Theorem 3. Given a set $\{FCPN_1, \ldots, FCPN_n\}$ of FCPNs derived from the ECTS *TS*, there is a bisimulation B such that $TS \sim_B ||_{i=1,\ldots,n} RG(FCPN_i)$ iff:

1) the set $\{FCPN_1, \ldots, FCPN_n\}$ of FCPNs satisfies *excitation closure* over the events of *TS*;
2) each component satisfies *place-connectedness*.

Proof. See the appendix. □

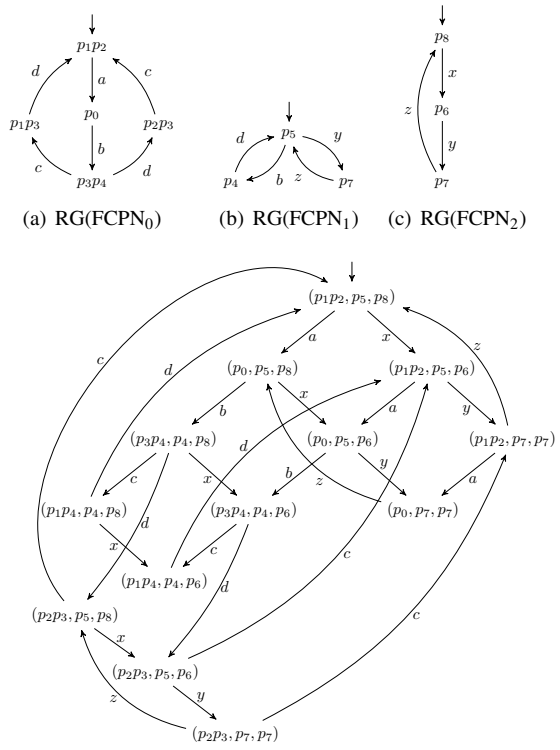
In Fig. 7 we can observe the reachability graphs of the FCPNs of our running example and their synchronous product which is bisimilar to the original ECTS.

Notice that our proof is constructive, i.e. it builds the bisimulation relation by defining the structural mapping. One could take a behavioural approach showing a relation between languages using the theory in [11] (which implies the existence of a bisimulation for deterministic systems), but this would not yield the actual bisimulation.

V. Experimental results

We implemented the procedure described in Sec. III, writing a software in C++ with dependencies to *PBLib* [12], library for the encoding and resolution of SAT formulas. We used the same transition systems of [6] and the couple *pparb_2_3* and *pparb_2_6* from [13]. Behaviors without concurrency were excluded from the set of benchmarks because they cannot take advantage of FCPN decomposition. Benchmark statistics were reported in Table III.

Table IV reports the numbers of SMs derived in [6] and of FCPNs derived by our procedure, the number of places after each decomposition step ("Decomp." column represents Alg. 1 without performing PetriNetSimplification(FCPN), "Minim." after algorithm in Sec. III-B, "Merge" after merging algorithm presented in Sec.III-C.) and the total computation time. It shows that in many cases FCPNs are fewer than SMs due to



(a) RG(FCPN$_0$) (b) RG(FCPN$_1$) (c) RG(FCPN$_2$)

(d) RG(FCPN$_0$)||RG(FCPN$_1$)||RG(FCPN$_2$)

Fig. 7: Reachability graphs of FCPN of the running example and their synchronous product.

	States	Transitions	Events	Min regions
alloc-outbound	17	18	14	15
clock	10	10	4	11
dff	20	24	7	20
espinalt	27	31	20	23
fair_arb	13	20	8	11
future	36	44	16	19
intel_div3	8	8	4	8
intel_edge	28	36	6	27
isend	53	66	15	128
lin_edac93	20	28	8	10
master-read	8932	36226	26	33
pe-rcv-ifc	46	62	16	7
pparb_2_3	1088	3392	22	45
pparb_2_6	96632	321536	34	75
pulse	12	12	6	33
rcv-setup	14	17	10	11
vme_read	255	668	26	44
vme_write	821	2907	30	51

TABLE III: Transitions systems used in the experiments.

the concurrency exhibited by FCPNs. There is still a number of results with as many FCPNs as SMs (if so, usually FCPNs coincide with the SMs). As to the impact on size of each optimization step, we notice that the average decrease of places and transitions after the optimizations performed during the decomposition process is higher than 15% showing the utility of these additional steps. Since the number of FCPNs is not directly shown, one can deduce that at least one FCPN was removed during the greedy FCPN removal by comparing the number of places after this minimization step to the initial

	# components		Total FCPN places after			Runtime[3]
	SMs	FCPNs	Decomp.	Minim.	Merge	(s)
alloc-outbound	2	2	21	21	17	<1s
clock	3	2	11	11	10	<1s
dff	3	3	50	38	33	<1s
espinalt	3	2	34	34	27	<1s
fair_arb	2	2	12	12	12	<1s
future	3	2	42	30	19	<1s
intel_div3	2	2	12	12	10	<1s
intel_edge	4	4	53	53	43	2.49
isend	13	5	138	97	74	68.50
lin_edac93	3	2	14	14	11	<1s
master-read	8	1	33	33	33	8.71
pe-rcv-ifc	2	2	49	49	43	8.98
pparb_2_3	11	2	46	46	46	2.46
pparb_2_6	18	3	86	86	83	658.46
pulse	2	2	7	7	7	<1s
rcv-setup	2	1	11	11	11	<1s
vme_read	9	5	97	97	74	2.96
vme_write	11	4	88	88	71	4.02
Total	101	46	804	739	624	
Average	5.6	2.6	44.7	41.1	34.7	
Normal. avg.	1.0	0.46	1.0	0.92	0.78	

TABLE IV: Results of the decomposition.

number of places.

From the computation times reported in Table IV, the benchmarks *isend* and *pparb_2_6* are two outliers, explained by the fact that they have many minimal regions, since the generation of regions is the bottleneck of the overall computation process accounting on average for more than 90% of it. In addition, the size of the regions matters: indeed the computation time of *pparb_2_6* is an order of magnitude more than the one of *isend*, even though *pparb_2_6* has half the regions of *isend*, since from Table III the regions in *pparb_2_6* are much larger than the ones of *isend*. Despite of the bottleneck to compute regions, our approximate algorithm in Sec. III can handle quite large transition systems. The result is not guaranteed to be a minimum one, but the irredundancy procedure guarantees a form of local minimality that guarantees the creation of free-choice Petri nets.

VI. Conclusion

In this paper we described a new method for the decomposition of transition systems. Our experimental results demonstrate that the decomposition algorithm can be run on large transition systems. The resultant set of synchronizing FCPNs offers a good complexity trade-off compared to other more expressive classes of PNs [4] vs. decomposition into completely sequential SM, as Fig. 2 suggests.

Since the generation of minimal regions is currently a computational bottleneck, future work will address this limitation, while it will leverage the improvements in efficiency of last-generation SAT solvers, and the power of HPC since the generation of minimal regions is highly parallelizable. As future work, we want also to apply this decomposition paradigm to process mining.

References

[1] E. Badouel, L. Bernardinello, and P. Darondeau, *Petri net synthesis*. Berlin: Springer, 2015.

[3]Runtime includes all phases of the decomposition.

[2] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev, "Deriving Petri nets from finite transition systems," *IEEE Transactions on Computers*, vol. 47, no. 8, pp. 859–882, Aug 1998.
[3] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on information and Systems*, vol. 80, no. 3, pp. 315–325, 1997.
[4] J. Esparza, "Decidability and complexity of Petri net problems—an introduction," in *Advanced Course on Petri Nets*. Springer, 1996, pp. 374–428.
[5] A. Ehrenfeucht and G. Rozenberg, "Partial (set) 2-structures," *Acta Informatica*, vol. 27, no. 4, pp. 343–368, 1990.
[6] V. Teren, J. Cortadella, and T. Villa, "Decomposition of transition systems into sets of synchronizing state machines," in *2021 24th Euromicro Conference on Digital System Design (DSD)*. IEEE, 2021, pp. 77–81.
[7] W. M. Van der Aalst, V. Rubin, H. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther, "Process mining: a two-step approach to balance between underfitting and overfitting," *Software & Systems Modeling*, vol. 9, no. 1, p. 87, 2010.
[8] J. Carmona, J. Cortadella, and M. Kishinevsky, "Divide-and-conquer strategies for process mining," in *International Conference on Business Process Management*. Springer, 2009, pp. 327–343.
[9] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.
[10] E. Boros and P. L. Hammer, "Pseudo-Boolean optimization," *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 155–225, 2002.
[11] A. Mazurkiewicz, "Compositional semantics of pure place/transition systems," *Fundamenta Informaticae*, vol. 11, no. 4, pp. 331–355, 1988.
[12] T. Philipp and P. Steinke, "PBLib – A Library for Encoding Pseudo-Boolean Constraints into CNF," in *Theory and Applications of Satisfiability Testing – SAT 2015*, ser. Lecture Notes in Computer Science, M. Heule and S. Weaver, Eds. Springer International Publishing, 2015, vol. 9340, pp. 9–16.
[13] V. Khomenko, M. Koutny, and A. Yakovlev, "Detecting state encoding conflicts in STG unfoldings using SAT," *Fundamenta Informaticae*, vol. 62, no. 2, pp. 221–241, 2004.

APPENDIX

A. Proof of theorem 1

Proof. Suppose that r_1 and r_2 are non-disjoint regions and $r_1 \cap r_2$ is also a region. Then $r_1 \setminus (r_1 \cap r_2) = r_1 \setminus r_2$ is a region, since r_1 and $r_1 \cap r_2$ are regions and $(r_1 \cap r_2) \subseteq r_1$ (by Prop. 1.50, p. 43, [1]). Then $r_1 \setminus r_2$ and r_2 are two disjoint regions and therefore $(r_1 \setminus r_2) \cup r_2 = r_1 \cup r_2$ is a region, since the union of two disjoint regions is also a region (by Prop. 1.64, p. 50, [1]).

Suppose that r_1 and r_2 are non-disjoint regions and $r_1 \cup r_2$ is a region, then $(r_1 \cup r_2) \setminus r_1$ and $(r_1 \cup r_2) \setminus r_2$ are regions, as differences of regions. So $((r_1 \cup r_2) \setminus r_1) \cup ((r_1 \cup r_2) \setminus r_2)$ is a region as union of disjoint regions. Then $r_1 \cap r_2 = (r_1 \cup r_2) \setminus (((r_1 \cup r_2) \setminus r_1) \cup ((r_1 \cup r_2) \setminus r_2)))$ is a region as difference of regions. □

B. Proof of theorem 3

Place connectedness is a property which consists in keeping all connections of a region (*exit/enter*) passing from regions to the place representation in FCPNs. This fundamental property, previously left implicit, enables us to use regions in the bisimulation, as it will be described soon. The lack of *place connectedness* means the loss of connections passing from a region r to a place p. As result p does not represent r anymore, but it represents an invalid region or a different region r' which has a *no cross* relation (neither *enter* nor *exit*) with the missing event. A region could be represented by different instances of places in two or more different FCPNs, but each of these places still preserves the same connections to the events of the represented region.

In this proof the following nomenclature will be used:
R is the total set of regions
$R_i \subseteq R$ is the set of regions represented by places of *FCPN*$_i$
$R(s)$ is the set of regions that contain state s
$R_i(s) = R(s) \cap R_i$ is the set of regions represented by places of *FCPN*$_i$ that include s.
$\overline{R(s)} = (R_1(s), \ldots, R_n(s))$, this alias will be used for improved readability.

The equivalence between an ECTS and the derived set of FCPNs is proved by defining a bisimulation between the original TS, defined as $TS = (S, E, T, s_0)$, and the synchronous product of the reachability graphs of the derived free-choice Petri nets $RG(FCPN_1)||RG(FCPN_2)|| \ldots ||RG(FCPN_n)$, denoted by $||_{i=1,\ldots,n} RG(FCPN_i) = (S_{||},\ E, T_{||},\ s_{0,||})$. Notice that each $RG(FCPN_i) = (R_i, E_i, T_i, R_i(s_0))$, with $T_i \subseteq R_i \times E_i \times R_i$, is defined on a subset E_i of events of *TS*, its states R_i corresponds to regions or sets of regions derived from *TS* represented by the markings $[M\rangle$ of $FCPN_i$.

The initial state of $RG(FCPN_i)$ is represented by $R_i(s_0)$: one or more regions containing the initial state s_0 of TS.

Another fundamental property, is the *excitation closure*. Indeed to prove the existence of a bisimulation we require that the regions contained in the set of FCPNs satisfies EC, where event-effectiveness guarantees that $\cup E_i = E$. Excitation closure property is crucial to prove the steps 1) and 3) of the proof.

Proof. We define the binary relation B as follows:

$$(s, \overline{R(s)}) \in B \iff s \in \bigcap_{i=1}^{n} r \mid r \in R_i(s) \qquad (1)$$

where:

- $s \in S$;
- $\overline{R(s)}$ or $(R_1(s), \ldots, R_n(s))$ is a tuple of markings where each marking $R_i(s)$ contains s.

Notice that writing $(s, \overline{R(s)}) \in B \iff \{s\} = \bigcap_{i=1}^{n} r \mid r \in R(s)$ would be wrong, because the intersection of regions could have two or more bisimilar (i.e., behaviourally equivalent) states, as in the TS $s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{a} s_3 \xrightarrow{b} s_0$.

Now we prove that B is a bisimulation in three steps:

1) $(s_0, \overline{R(s_0)}) \in B$.
2) If $(s_j, \overline{R(s_j)}) \in B$ and $(s_j, e, s_k) \in T$, then there is $\overline{R(s_k)} \in S_{||}$ such that $(\overline{R(s_j)}, e, \overline{R(s_k)}) \in T_{||}$ and $(s_k, \overline{R(s_k)}) \in B$.
3) If $(s_j, \overline{R(s_j)}) \in B$ and $(\overline{R(s_j)}, e, \overline{R(s_k)}) \in T_{||}$, then there is $s_k \in S$ such that $(s_j, e, s_k) \in T$ and $(s_k, \overline{R(s_k)}) \in B$.
4)

Let us now proceed with the proofs.

1) Since *TS* has a unique initial state s_0, each FCPN *FCPN*$_i$ has at least one initial region $r \in R_i(s_0)$ such that $s_0 \in r$ because the regions of an FCPN cover all the states of initial TS (otherwise *excitation closure* wouldn't be satisfied). Therefore, $s_0 \in \bigcap_{i=1}^{n} r \mid r \in R_i(s_0)$ and we have that $(s_0, \overline{R(s_0)}) \in B$.

2) Since $(s_j, e, s_k) \in T$ and $(s_j, \overline{R(s_j)}) \in B$, then $s_j \in \bigcap_{i=1}^{n} r \mid r \in R_i(s_j)$. Now we will prove that there is s_k such that $s_k \in \bigcap_{i=1}^{n} r \mid r \in R_i(s_k)$, so that we can have $(s_k, \overline{R(s_k)}) \in B$.

Notice that FCPN$_i$ may share places, but the property of place-connectedness guarantees that shared places have the same marking, i.e. if place p occurs in more than one FCPN$_i$ and an event e is entering or exiting in one occurrence of p then e will be entering or exiting in all occurrences of p.

Since e is enabled in s_j, no region $R_i(s_j)$ can be a post-region of e. If one $r \in R_i(s_j) \in \overline{R(s_j)}$ would be a post-region, then $s_j \notin \bigcap_{i=1}^{n} r \mid r \in R_i(s_j)$. Therefore, the following cases can be distinguished for each $r \in R_i(s_j) \in \overline{R(s_j)}$:

- e is not an event of *FCPN$_i$*. Thus, $R_i(s_k) = R_i(s_j)$.
- e is an event of *FCPN$_i$*,
  - and all regions $r \in R_i(s_j)$ are no-cross regions for e. Thus, $R_i(s_k) = R_i(s_j)$. Explanation: say that each of the n places of the current marking $R_i(s_j)$ represents a region. By place-connectedness each place is connected to all events of the corresponding region. So, if e is no-cross with respect to all regions, then no place of the FCPN will be connected to e, and firing e cannot remove tokens from any place of $R_i(s_j)$, and so $R_i(s_k) = R_i(s_j)$.
  - and all regions $r \in R_i(s_j)$ are pre-regions of e. Thus, $R_i(s_k) \neq R_i(s_j)$ and each region $r \in R_i(s_k)$ is a post-region of e. By place-connectedness.
  - and some regions of $R_i(s_j)$ are pre-regions of e. Thus, $R_i(s_k) \neq R_i(s_j)$ and each region $r \in R_i(s_k) \setminus R_i(s_k)$ is a post-region of e. By place-connectedness.

For the first and second case, *FCPN$_i$* will not change marking and *TS* will not change region when moving from s_j to s_k. Therefore, $s_k \in r \in R_i(s_j) = R_i(s_k)$.

For the third and fourth case, e will exit $r \in R_i(s_j)$ in at least one FCPN and will enter $r \in R_i(s_k)$ in *TS*, which means that $s_k \in r \in R_i(s_k)$. Therefore, $(\overline{R(s_j)}, e, \overline{R(s_k)}) \in T_{||}$.

For all cases we have that $s_k \in r \in R_i(s_k)$ and therefore $s_k \in \bigcap_{i=1}^{n} r \mid r \in R_i(s_k)$.

3) Since $(s_j, \overline{R(s_j)}) \in B$, it holds that $s_j \in \bigcap_{i=1}^{n} r \mid r \in R_i(s_j)$. Given the existence of the transition $(\overline{R(s_j)}, e, \overline{R(s_k)})$, and knowing that the *excitation closure* property holds, we know that $s_j \in \bigcap_{i=1}^{n} r \mid r \in R_i(s_j) \subseteq ES(e)$. The latter inequality holds because we have 1) by PN construction, $\forall i, i = 1, \ldots, n$, label e appears once in *FCPN$_i$* or it does not appear, and 2) being all states covered by at least one region of the set of FCPNs, $\forall i, i = 1, \ldots, n$, if label e appears in *FCPN$_i$* then $r \in R_i(s_j)$ could be a no-cross region, otherwise $r \in R_i(s_j) \in ({}^\circ e \cap R)$, by which $\bigcup_{i=1}^{n} r \mid r \in R_i(s_j) \supseteq \bigcup_{r \in ({}^\circ e \cap R)} \{r\}$ and so by intersection of the regions seen as sets of states $\bigcap_{i=1}^{n} r \mid r \in R_i(s_j) \subseteq \bigcap_{r \in ({}^\circ e \cap R)} r = ES(e)$.
Therefore, there is s_k such that $(s_j, e, s_k) \in T$. We can also see that $s_k \in \bigcap_{i=1}^{n} r \mid r \in R_i(s_k)$, using the same reasoning as in step 2, since all the pre-regions $r \in R_i(s_j)$ of e in $\overline{R(s_j)}$ are exited by entering $r \in R_i(s_k)$, whereas the no-crossing regions remain the same. We can then conclude that $(s_k, \overline{R(s_k)}) \in B$. □