Seto: a framework for the decomposition of Petri nets and transition systems

Viktor Teren
Università degli Studi di Verona
viktor.teren@univr.it

Jordi Cortadella
Universitat Politècnica de Catalunya
jordi.cortadella@upc.edu

Tiziano Villa
Università degli Studi di Verona
tiziano.villa@univr.it

***Abstract*—This paper presents an overview of different approaches, based on theory of regions, for Transition System and Petri net decomposition into a synchronous product of restricted subclasses of Petri nets. A decomposition targeting State Machines was implemented in a prototype software, by reduction to maximal independent set which computes minimal sets of irredundant state machines. Then, states of single state machines were merged by reduction to the Boolean satisfiability problem (SAT). Furthermore, an extension to Free-choice Petri net decomposition was implemented, reducing the whole decomposition process to a series of SAT problems. We report experimental results that show a good trade-off between quality of results vs. time of computation, including different variants of the decomposition flows. In particular, we introduce a new approach allowing a simultaneous search of components, exploiting Binary Decision Diagrams and the excitation-closure property provided by theory of regions.**

I. Introduction

This paper presents Seto[1] a framework for the decomposition of Labeled Transition Systems (LTSs) and Petri nets (PNs) into a synchronous product of State Machines (SMs) or Free-choice Petri nets (FCPNs). The bridge that enables the decomposition process is the theory of regions [1] already used for decades in PN synthesis. The theory of regions allows not only to create a versatile decomposition framework, but also to prove the existence of a bisimulation between the set of derived synchronized SMs [2] and FCPNs [3] and the initial LTS or PN. In the next sections we will present the new improvements introduced in the decomposition flows presented in [3], [4]. The main contributions of this paper are:

- A simultaneous decomposition flow with a heuristic and an optimal version of the algorithm: a new approach based on BDDs and the theory of regions (Sec. VII);
- A method to ensure only safe FCPNs as result of a decomposition (Sec. VIII).

A. Overview

The LTS decomposition can be used in multiple ways. The most intuitive contribution is to obtain a compact representation of highly concurrent behaviors, with explicit disentanglement of key components, as in the case of Figs. 1 and 2. This example shows a clear distinction between two parts of the behavior, with alphabets $\{a, b, c, d\}$ and $\{x, y, z\}$, respectively, shown at the left (a) and right (c) of the picture and an arbitration process (b) that creates a mutual exclusion between $\{b, d\}$ and $\{y, z\}$. The benefit of the decomposition can also be valued when a single PN represents a "spaghetti" structure, very difficult to analyze.

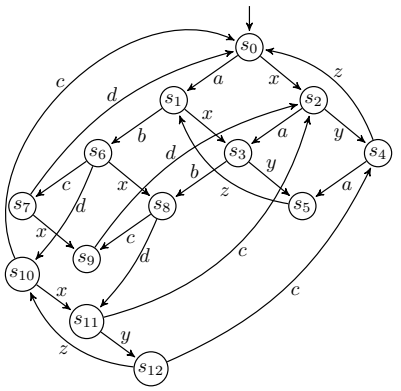
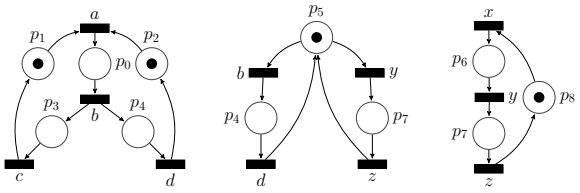


Fig. 1: Transition System.



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

Fig. 2: Three FCPNs distilled from the LTS in Fig. 1

The power of the decomposition process consists also in the discovery of hidden behaviors like in the case of Fig. 3, representing an apparently sequential behavior which is hiding different concurrent flows, indeed the FCPN decomposition of ECTS in Fig. 3 is represented by two components (Fig. 4). This kind of property plays an important

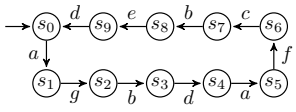
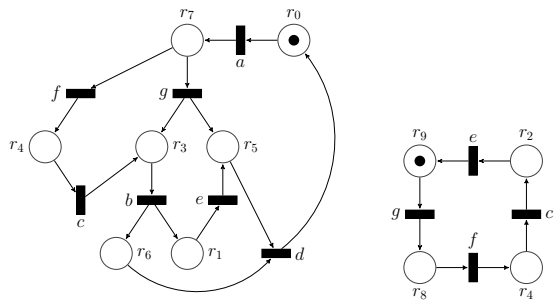


Fig. 3: ECTS.

[1]available at https://github.com/viktorteren/Seto

2771-2508/23/$31.00 ©2023 IEEE
DOI 10.1109/DSD60849.2023.00096



(a) FCPN$_0$ (b) FCPN$_1$

Fig. 4: FCPNs derived from LTS in Fig. 3.

role in the process mining environment, indeed, decomposition has already an important place in process mining, especially in Business Process Management (BPM) [5]–[8]. Instead of creating a PN from event logs, we could easily create an LTS [9], [10] and directly decompose it with our algorithm.

The purpose of our algorithm is to provide a new exploration framework to work in synergy with established process mining technologies, without replacing them.

B. State-of-the-art

Referring to PN and LTS decomposition, in [11], an LTS is decomposed iteratively into an interconnection of n component LTSs with the objective to extract a PN from them. In [11] the decomposition allows the extraction of a PN, but the decomposed set of LTSs cannot be used as an intermediate model. This approach is flexible in choosing how to split the original LTS, but it does not provide any minimization algorithm, so that the redundancy due to overlapping states in the component LTSs translates into redundant places of the final PN. Another method presented in [12] is based on the decomposition of LTSs into *"slices"*, where each LTS is separately synthesized into a PN, and in case of PNs "hard" to understand the process can be recursively repeated on one or more *"slices"* creating a higher number of smaller PNs. With respect to the aforementioned methods, our approach yields by construction a set of PNs restricted to the desired subclass and applies to them minimization criteria. The results of [13] instead show how complex processes can be formally represented by process windows, where each window covers a part of process behavior. In our case, each PN could be interpreted as a window representing a part of the entire process.

II. Preliminaries

This section provides the background material of the paper.

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

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

Henceforth, $s \xrightarrow{e} s'$ will denote the fact that $(s, e, s') \in \Delta$. Every LTS is assumed to meet the following properties:

- It does not contain-self loops, i.e., $s \xrightarrow{e} s' \implies s \neq s'$.
- Each event has at least one occurrence, i.e., $\forall e \in E : \exists s \xrightarrow{e} s'$.
- Every state $s \in S$ is reachable from the initial state, i.e., there is a sequence $s_0 \xrightarrow{e_1} s_1, s_1 \xrightarrow{e_2} s_2, \ldots, s_{n-1} \xrightarrow{e_n} s_n$ such that $s_n = s$.
- It is deterministic, i.e., $s \xrightarrow{e} s', s \xrightarrow{e} s'' \implies s' = s''$.

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

The operation *Ac* deletes from an LTS 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 LTSs $TS_1 = (S_1, E_1, T_1, s_{0,1})$ and $TS_2 = (S_2, E_2, T_2, s_{0,2})$, the synchronous product is defined as

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

where $S \subseteq S_1 \times S_2$, $(s_{0,1}, s_{0,2}) \in S$, $T \subseteq (S_1 \times S_2) \times E \times (S_1 \times S_2)$ 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 synchronous product is associative, so we can extend the previous definition to more than two LTSs.

Definition 4 (Ordinary Petri net, PN [14])**.** An ordinary Petri net is a 4-tuple, $N = (P, T, F, M_0)$ 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),
- $M_0 : P \to \{0, 1, 2, 3, ...\}$ is an initial marking,
- $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

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

Definition 5 (Firing rule)**.** Let $N = (P, T, F, M_0)$ be a PN. A transition $t \in T$ is enabled in marking M, represented as $M[t\rangle$, if $M(p) > 0, \forall p \in {}^\bullet t$. If t is enabled in M, then t can

be fired leading to another marking M', denoted as $M[t\rangle M'$, such that:

$$M'(p) = \begin{cases} M(p) - 1 & \text{if } p \in {}^\bullet t \backslash t^\bullet \\ M(p) + 1 & \text{if } p \in t^\bullet \backslash {}^\bullet t \\ M(p) & \text{otherwise} \end{cases}$$

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

Definition 6 (Reachability graph [15, p. 20])**.** Given a safe PN $N = (P,\ T, F, M_0)$, the reachability graph of N is the LTS $RG(N) = ([M_0\rangle, T,\ \Delta, M_0)$ defined by $(M, t, M') \in \Delta$ if $M \in [M_0\rangle$ and $M[t\rangle M'$.

Definition 7 (State Machine, SM [14])**.** A State Machine is a Petri net $N = (P, T, F, M_0)$ such that each transition $t \in T$ has exactly one incoming and one outgoing arc, i.e., $|{}^\bullet t| = |t^\bullet| = 1$.

An SM with only one token on the initial marking cannot model concurrency, but it can model choice.

Definition 8 (Free-choice Petri net, FCPN [14])**.** 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 $p_1, p_2 \in P$, $p_1^\bullet \cap p_2^\bullet \neq \emptyset \Rightarrow |p_1^\bullet| = |p_2^\bullet| = 1$.

III. Regions: the bridge between Transition System and Petri net

In this paper we propose a procedure for the decomposition of Transition Systems based on the theory of regions.

Definition 9 (Region)**.** Given an LTS $TS = (S, E,\ T, s_0)$, a region is defined as a non-empty set of states $r \subsetneq S$ such that the following properties hold for each event $e \in E$:

$$\begin{aligned} enter(e,r) &\implies \neg in(e,r) \wedge \neg out(e,r) \wedge \neg exit(e,r) \\ exit(e,r) &\implies \neg in(e,r) \wedge \neg out(e,r) \wedge \neg enter(e,r) \\ no\_cross(e,r) &\implies \neg enter(e,r) \wedge \neg exit(e,r) \end{aligned}$$

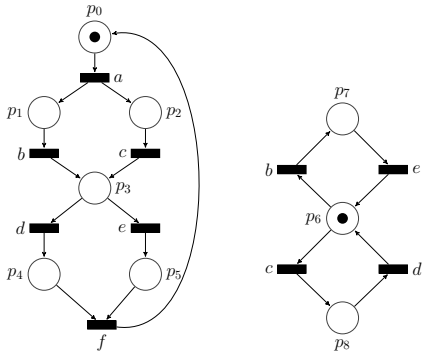
where

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

Definition 10 (Minimal region)**.** A region r is called *minimal* if there is no other region r' strictly contained in r ($\nexists r' \mid r' \subset r$).

Definition 11 (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 from r (enters into r). The set of all pre-regions (post-regions) of the event e is denoted by ${}^\circ e$ (e°).

If the transition system is strongly connected all the regions are also pre-regions.



(a) $FCPN_0$ (b) $FCPN_1$

Fig. 5: Two FCPNs with $FCPN_0$ unsafe.

Definition 12 (Excitation set)**.** The *excitation set* of event e, $ES(e)$, is the maximal set of states such that for every $s \in ES(e)$ there is a transition $s \xrightarrow{e}$.

Definition 13 (Excitation-closed Transition System, ECTS)**.** An LTS 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: $\forall e \in E : {}^\circ e \neq \emptyset$

If the initial LTS does not satisfy the excitation-closure or event effectiveness property, *label splitting* [1] can be performed to obtain an ECTS.

IV. Safe composition of unsafe Free-choice Petri nets

Given a set of FCPNs from a decomposition, even if the composition of FCPNs is safe, single components considered without synchronizations could be unsafe, still preserving the synchronous product of the reachability graphs with the original ECTS. Fortunately, in [16] the authors proved the existence of a safe PN starting from an ECTS. In our case the decomposition into sets of FCPNs can be seen as a special case. The decomposition into a set of synchronized SMs can be interpreted as the S-covering of a PN, which is proven to be safe. For FCPNs the existence of a safe SM decomposition implies also the existence of an FCPN decomposition, where in the worst case all components would be SMs.

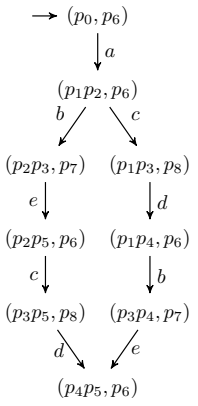


Fig. 6: Synchronous product of the reachability graphs of FCPNs in Fig. 5.

In the next sections, we will see different ways to decompose an LTS into SMs and FCPNs. In case of SMs, it is sufficient to avoid multiple initial places to keep safe an SM. For FCPNs, a direct safeness check will be needed.

Let's see an example of a safe composition with unsafe PNs. In Fig. 5 we have one unsafe FCPN. Possible sequences which bring to an unsafe marking in $FCPN_0$ could be "abc" and "acb". Even if $FCPN_0$ is unsafe, the synchronization with $FCPN_1$ makes unreachable all unsafe markings. Indeed, Fig. 6 represents the composition of the reachability graphs of the PNs, which never reaches an unsafe marking.

V. Decomposition into a set of synchronized Free-choice Petri nets or State Machines

As we have seen in Section I-A, there are plenty of different methods to perform a decomposition, our framework performs it relying on the theory of regions, therefore, given a set of SMs or 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 ${}^\bullet e$ of the *TS* for all $e \in E$, the set of events E_k and regions R_k of PN k, to guarantee the existence of a bisimulation between the initial model and the resultant set of synchronizing components, the following properties must be satisfied:

- *Excitation-closure*: $\forall e \in E : \bigcap_{r \in ({}^\bullet e \cap R)} r = ES(e)$
- *Event effectiveness*: $\forall e \in E : \exists r \in R \mid r \in {}^\bullet e$
- *Place-connectedness*:
$$\forall r \in R : (r \in R_k, e_i \in ({}^\bullet r \cup r^\bullet)) \implies e_i \in E_k$$

The first property guarantees that for each event of the original PN there is a sufficient number of its pre-regions across the derived PNs, so that, given a marking in the decomposed set of PNs, a transition representing the event can fire iff a transition with the same event fires in the original PN. An insufficient number of pre-regions of an event would relax the constraint allowing the activation of events whose activation is not allowed in the original PN.

The second property guarantees that for each event of the original PN there is a region from which the event is enabled to fire.

The third property guarantees that, if a region represents a place in one of the derived PNs, this place will keep the connections of the region to all events for which the region is a pre-region or a post-region.

The main steps to obtain a set of synchronizing SMs or FCPNs, reported in [3], [17], are:

1) Transformation of the PN into an LTS: reachability graph extraction
2) If LTS is not an ECTS *label splitting* is performed
3) Computation of all minimal regions
4) Generation of a set of SMs or FCPNs with *EC* property
5) Optional: removal of redundant PNs
6) Optional: merge between regions preserving EC

The first three steps of the flow can be easily found in literature and in the section on preliminaries. For the other steps, instead, some methodologies are shown in this section.

In order to decompose an LTS/PN in a set of synchronized PNs the excitation-closure property has to be maintained, since it is necessary and sufficient for a correct behavior of the decomposed PN. The decision on how to choose the regions for each component may affect the size of the result, but if it is excitation-closed it will preserve the original behavior. The result can have a lot of redundancy, especially because the most efficient flows rely on heuristic steps and pass through a Saturated Petri net (SPN), which describes the maximally redundant set of places. For this reason, after the decomposition, redundant PNs can be removed. In some cases it is not possible to remove entire PNs but only redundant parts of them. For a further insight on the optimization steps we refer to [3], [17].

There is not only one way to generate a set of synchronizing PNs from an LTS, especially if we restrict the resultant PNs to different subclasses.In the next sections the following approaches will be presented for the decomposition into sets of synchronizing SMs or FCPNs:

- Sequential search: based on an approximated method, obtains really scalable and suitable results.
- Simultaneous search: guarantees the minimality of the number of PNs but, as seen in the following sections, the performance depends a lot on the desired PN subclass decomposition.

Approximated methods were exploited since an exact algorithm would not be suitable, with runtime and memory limits reachable with around 15 regions, when a standard benchmark could have more than one hundred regions (see Table I).

VI. Decomposition approaches previously adopted by our framework

Both SM and FCPN decomposition flows share the same first steps available in literature, including the generation of the minimal regions by a greedy algorithm, which checks minimality while creating regions [1] [15, p. 103] [18].

A. Sequential generation of State Machines with Excitation-closure property

The generation of a set of SMs is performed by reducing it to an instance of maximal independent set (MIS)[2], and by calling a MIS solver on the graph whose vertices correspond to the minimal regions with edges connecting regions that intersect. Each MIS of the aforementioned graph corresponds to a set of disjoint regions that define an SM. The detailed algorithm can be found in [2].

The set of derived SMs satisfies the EC property because by construction each region is included in at least one independent set, therefore starting from the set of all minimal regions the condition $\forall e \in E : \bigcap_{r \in {}^\circ e} r = ES(e)$ is satisfied, otherwise, if there would be an event which does not satisfy the EC, the initial LTS would not be an ECTS either. Also, event effectiveness holds because, starting from all minimal regions of the initial ECTS, for each event e there is at least one region $r \in {}^\circ e \neq \emptyset$ which is also in at least one SM. Finally, the place connectedness property also holds because the choice of a MIS does not allow to split two or more different paths outgoing from a region, because alternative paths will always

[2]Given an undirected graph $G = (V, E)$, an **independent set** is a subset of nodes $U \subseteq V$ such that no two nodes in U are adjacent. An independent set is maximal if no node can be added without violating independence.

have independent regions and therefore all these regions will be included in the same MIS result.

Sometimes a set of regions given from the MIS solver could actually represent two (or more) SMs, since no constraint on the complete connection of all regions has been added. That is not a problem, since we can split the disconnected sets of regions as separate SMs forbidding the smallest ones. In this way sooner or later a solution with only one SM will be found and it will be the largest available SM. A solution with smaller SMs would be valid, but heuristically without these small SMs a solution with a minor number of SMs would be more likely to be found.

B. Sequential Free-choice Petri net search

In this case as a starting point we could take the flow presented for SMs, which already satisfy also the constraints for the FCPNs, but we could do even better: FCPNs are less restrictive than SMs allowing concurrency inside the components, therefore we could take advantage from the previously presented flow, adopting some changes. In addition to *excitation-closure*, *event effectiveness* and *place connectedness*, we show the properties which must be satisfied when a set of regions R_k represents an FCPN:

- *Event connectedness*. If an event e is present, at least one pre-region and one post-region of e must belong to R_k.
- *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 $r \in R_k$, $|r^\bullet| > 1$, $e \in r^\bullet$, $r' \in {}^\bullet e$, and $r \neq r'$, then $r' \notin R_k$.

In order to create a restricted set of FCPNs also the following constraints have to be taken into account during the creation of a new FCPN:

- *Maximization function*: given the set of minimal regions, during the creation of a new FCPN, maximize the usage of regions not included in any of the previously created FCPNs;
- (Optional) *Minimization function*: minimize the number of the total used regions in the new FCPN keeping constant the maximum number of new used regions.

The purpose of the first functions is to create as few FCPNs as possible: the maximization function forces the creation of the largest possible FCPNs, so that the number of regions is sufficient to achieve a set of synchronized FCPNs can be reached reducing the number of used FCPNs. The second function instead is used in order to minimize the number of redundant regions during the creation of a new FCPN, keeping constant the result obtained by the previous constraint.

Looking into the implementation of this kind of decomposition, differently from the SMs, in this case, in order to find a set of synchronized FCPNs a SAT solver is used. Each of the previously described properties was encoded into a set of CNF clauses, then, using binary search in the range $[0, n]$, instances of SAT problems were solved trying to find a new FCPN with $i \in [0, n]$ new regions, where the maximum number of unused regions is n. Afterward, keeping fixed the number of obtained FCPNs, single FCPN minimization can be performed (merging algorithm).

VII. New decomposition approach implemented in our framework

A. Decomposition into k State Machines/Free-choice Petri nets simultaneously

In this paper, we present a novel method for the decomposition of LTSs into sets of synchronized SMs/FCPNs. This method has a lot of similarities to the method used for the sequential FCPN search, but the approach is different since the search is performed simultaneously on k components. This is allowed by the encoding of excitation-closure property, using the set of minimal regions obtained from the LTS. In order to encode the excitation-closure, for each event, all possible sets of minimal regions that satisfy EC are found. The complexity of this step is $O(2^n)$, where n is the number of pre-regions of the event. Fortunately, the maximum number of pre-regions for an event hardly exceeds a dozen. Once all possible sets of pre-regions which satisfies EC for each event were found, these sets were encoded into binary decision diagrams (BDDs), using the CUDD package [19]. Each BDD represents a set of choices of regions in order to satisfy excitation-closure for the given event. It is important to notice that each BDD is composed by pre-regions for the given event and not all regions of an LTS. The path from the root to a leaf represents a set of choices of the pre-regions of the given event, which can bring to the satisfaction of excitation-closure or not. Taking all paths that satisfy EC, a set of DNF clauses which represents the EC satisfaction can be created. In our case, we need them in CNF form to be fed to a SAT solver, therefore the complement is taken, i.e., all paths that do not satisfy EC. In detail: a path root-leaf can be represented as a clause where each literal is a region which could be *true* or *false*, i.e., according to whether the region is taken or not, for example $(r_1 \wedge r_2 \wedge \overline{r_3})$ represents a set of three pre-regions for an event where r_1 and r_2 are taken and r_3 not. This path could lead to an unsatisfied excitation-closure, as many other paths could do. For the satisfaction of the EC property, it is sufficient to avoid each of these unsatisfiable paths. Given the set of clauses C where each clause represents a set of region assignments that bring to unsatisfied excitation-closure, a set of clauses $\bigwedge_{c_i \in C} \overline{c_i}$ is created to represent that each clause $c_i \in C$ has to be avoided. Being each clause c_i a conjunction of literals, $\overline{c_i}$ becomes a disjunction of literals, bringing the entire set of clauses into CNF form. The aforementioned method works with any number of clauses, but we can do even better, noticing that sometimes there is a special case with only one set of regions satisfying excitation-closure, in which case not all possible unsatisfiable paths should be visited, since a set of CNF clauses representing the usage of all involved regions can be directly created. For example, if for a given event e two pre-regions r_1 and r_2 are sufficient to ensure EC, it is possible to create a set of two clauses: $(r_1) \wedge (r_2)$.

Combining the new EC encoding procedure with the previously presented decomposition flow we obtain the following set of constraints which has to be satisfied:

1) *Excitation-closure*;
2) *Place connectedness*: if a region r is used in an FCPN, then all events with incoming or outgoing edges with respect to r must appear in it;
3) *Event connectedness*: structural connection between events and regions; if an event is taken, at least one pre-region and one post-region of this event should be in the SM/FCPN, this constraint guarantees structurally simple components;
4)
   - *FCPN structure*: prohibits symmetric choice, asymmetric choice and conflict structures;
   - *SM structure*: prohibits synchronization of parallel flows, symmetric choice, asymmetric choice and conflict structures;
5) (Optional) *Event usage*: each event has to be taken at least once: this constraint can be derived from the combination of constraints 1 and 2, but the usage of clauses with only one literal can speed up the computation.

Having encoded the excitation-closure into a set of clauses, also all remaining constraints for a set of k SMs/FCPNs can be encoded, and with $k = 1$ a linear search can start, increasing k until a SAT result is found. Also, in this case, once a SAT result is found, a minimization function can be used, minimizing the number of used regions, and keeping the result satisfiable with a fixed number of components.

As it has been seen for the sequential search, also in case of the simultaneous one, a satisfiable set of regions could actually represent two (or more) components. It means that given the solution of the simultaneous method, the disconnected sets of regions can still be divided as separate SMs/FCPNs, reaching a suboptimal result. In order to be sure that the achieved result is optimal, it is possible to sign the set of regions representing more than one component as a forbidden result, continuing the search for the optimal solution. This method is very simple, but it is not applied really often and does not produce a drastic performance drop.

If we performed the search for the optimal result there is no need for the greedy FCPN removal since we know that there is no solution with fewer FCPNs. Regarding the minimality of the final components, the simultaneous algorithm still needs a further merge of regions, as in case of the sequential versions.

In Fig. 7 is represented a flowchart combining the previous contributions and the newly introduced simultaneous decomposition with an in depth representation of the new contribution in Fig. 8.

VIII. Constraint for safe Free-choice Petri nets

Even if the safeness of single components is not necessary to have safe synchronization of FCPNs, a method to guarantee also the safeness of single components was proposed. This method works with the sequential FCPN search and consists in the addition of counters in order to address the search to safe FCPNs. The idea is very simple: every time an FCPN is found a check on safeness is performed, if the FCPN is unsafe, it is marked as forbidden and the search continues. Since without changing the previously presented sequential approach, the search would result "blind", here the counters

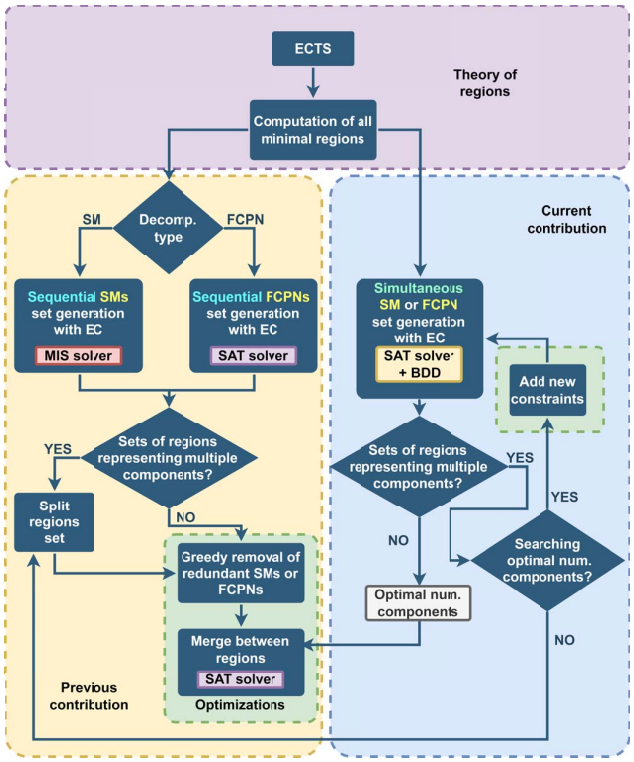


Fig. 7: Flowchart representing different flows implemented in our framework, highlighting the parts introduced in the previous versions and the one introduced in this paper.

enter into the picture. To improve the search, every time an unsafe FCPN is found, the counters are updated to show the regions which probably should be avoided in order to find a safe FCPN. How do these counters work? Each region has a dedicated counter, when an unsafe FCPN is found, the counter of the region is increased only if the region is a post-region of a fork transition, and the check is done only with respect to the set of regions appearing in the unsafe FCPN found as last. In the few cases with unsafe FCPNs, this method allows finding a set of safe FCPN within a reasonable time. The addition of the new constraint increases, both the decomposition time and the number of derived components. If the performance is very important at the expense of the size, an alternative can be the search of a set of SMs, which are also FCPNs by definition, since the provided algorithm natively extracts only safe components.

IX. Experimental results

For our tests, we used two sets of benchmarks, both from the world of asynchronous controllers: the first set (the same as in [20]), with smaller transition systems; the second one with large transition systems containing parametrized controllers (art_m_n) from [21] and two parametrized controllers from the set in [22]: "*pparb_2_3*" and "*pparb_2_6*". Basic statistics of the used benchmarks are reported in Table I.

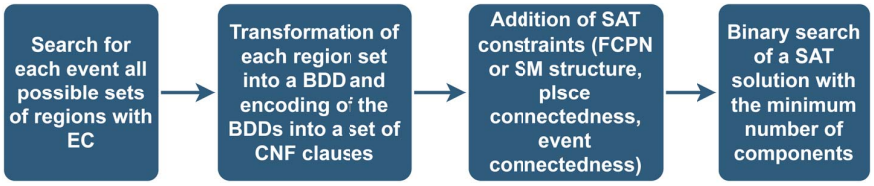


Fig. 8: The main steps of the simultaneous SM or FCPN set generation represented in Fig. 7.

The running environment consists in a computer with 4 cores and 8 threads CPU, 16 GB of RAM and Ubuntu 22.04.1 LTS. Our software is written in C++ with calls to *PBLib* [23] for the encoding and resolution of SAT formulas. The resolution of the MIS problem is performed by the *NetworkX* library [24], run on Python. The BDD encoding instead is performed by the CUDD library [19].

A. State Machines

We implemented the method presented in Sec. VII-A applying the SM structure constraint. In order to ensure only safe SMs, an additional constraint was added to limit to only one initially marked place, since multiple marked places could lead to unsafe markings. Table II compares the results with the sequential SM search. The new method can yield an almost minimal number of components, e.g., in the examples "*isend*", "*pparb_2_3*" and "*pparb_2_6*". In most cases, instead, the number of components remains unchanged. The average decomposition time spent is about five times higher with respect to the sequential approach, therefore the new method cannot directly compete with the previous one, but in the few cases with a reduced number of components the decomposition time is also reduced with respect to the sequential FCPN search. It means that it is still possible to run both methods in parallel, and probably the first method that completes will exhibit also the better result.

TABLE I: LTS statistics

Input	States	Transitions	Events	Regions
alloc-outbound	17	18	14	15
art_3_05	4000	11300	30	34
art_3_10	32000	93200	60	64
art_3_15	108000	317700	90	94
art_3_20	256000	756800	120	124
art_4_03	10368	37152	24	30
art_4_09	839808	3242592	72	78
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	36	26	33
pe-rcv-ifc	46	62	16	7
pparb_2_3	1088	3392	22	42
pparb_2_6	69632	321536	34	77
pulse	12	12	6	33
rcv-setup	14	17	10	11
seq_40	164	164	164	164
vme_read	255	668	26	44
vme_write	821	2907	30	51

TABLE II: Comparison between sequential SM search using previously created heuristics (sequential version) and the new approach (simultaneous version) directly encoding excitation-closure property.

Input	Number of derived SMs		Decomposition time [s]	
	sequential version	simultaneous version	sequential version	simultaneous version
alloc-outbound	2	2	0.12	0.03
art_3_05	5	5	0.36	0.04
art_3_10	5	5	2.26	8.29
art_3_15	5	5	10.51	45.22
art_3_20	5	5	31.30	146.27
art_4_03	7	7	0.63	1.62
art_4_09	7	7	62.98	456.23
clock	3	3	0.12	0.02
dff	3	3	0.25	0.18
espinalt	3	3	0.21	0.13
fair_arb	2	2	0.12	0.02
future	3	3	0.13	0.04
intel_div3	2	2	0.13	0.02
intel_edge	3	3	0.39	0.38
isend	7	5	0.68	0.40
lin_edac93	3	3	0.11	0.02
master-read	8	8	1.04	1.70
pe-rcv-ifc	2	2	0.35	0.31
pparb_2_3	12	7	0.56	0.28
pparb_2_6	18	15	41.24	47.02
pulse	2	2	0.12	0.02
rcv-setup	2	2	0.17	0.02
seq_40	1	1	0.75	1.42
vme_read	8	8	0.34	0.44
vme_write	10	10	0.75	1.00
AVERAGE	5.12	4.72	6.22	28.44

B. Free-Choice Petri nets

For the FCPN decomposition we used the same setup as in case of SM decomposition.

Table III shows the comparative results between the two main methods to extract FCPNs: sequential vs simultaneous decomposition and also the optimal version of the simultaneous approach. Differently from SMs, in case of FCPNs the simultaneous approach achieved good results: it was possible to dramatically decrease the average number of components keeping the decomposition time at the same level as before, even improving it a bit. Having performed also the optimal decomposition it is possible to see how the simultaneous version is close to the optimal result (1.96 FCPNs on average vs 1.92 FCPNs in the optimal case), and furthermore we can see that the average time to perform the optimal simultaneous decomposition requires more than twice the time required by the simultaneous version only to ensure the optimality

TABLE III: Comparison between sequential and simultaneous FCPN search.

Input	Number of derived FCPNs			Decomposition time [s]		
	sequential version	simultaneous version	simultaneous version (optimal)	sequential version	simultaneous version	simultaneous version (optimal)
alloc-outbound	2	2	2	0.03	0.03	0.03
art_3_05	2	1	1	0.64	0.46	0.44
art_3_10	2	1	1	10.11	8.50	9.87
art_3_15	2	1	1	45.42	43.94	45.23
art_3_20	2	1	1	151.09	143.00	1142.90
art_4_03	2	1	1	1.60	1.40	1.40
art_4_09	2	1	1	480.48	539.80	534.60
clock	2	2	2	0.01	0.02	0.01
dff	3	3	3	0.23	0.27	0.28
espinalt	2	2	2	0.07	0.06	0.06
fair_arb	2	2	2	0.01	0.02	0.02
future	2	2	2	0.01	0.02	0.02
intel_div3	2	2	2	0.01	0.02	0.02
intel_edge	3	3	3	0.41	0.47	0.39
isend	6	4	4	40.97	1.73	1.56
lin_edac93	3	2	2	0.01	0.01	0.01
master-read	2	1	1	1.85	1.42	1.53
pe-rcv-ifc	3	2	2	43.06	0.32	0.36
pparb_2_3	3	2	2	0.40	0.20	0.20
pparb_2_6	3	4	3	49.78	52.78	85.23
pulse	2	2	2	0.01	0.01	0.01
rcv-setup	1	1	1	0.02	0.01	0.01
seq_40	1	1	1	1.30	1.27	1.26
vme_read	3	3	3	0.45	0.46	0.49
vme_write	3	3	3	0.88	0.90	0.93
AVERAGE	2.40	1.96	1.92	33.15	31.88	73.07

(31.88 s vs 73.07 s). It is important to notice that the gap between the simultaneous version and the optimal one could grow exponentially with the increase of the number of regions used for the FCPN creation, taking even hours to perform the decomposition. As a conclusion, the simultaneous FCPN decomposition achieves the best trade-off between minimality of the number of derived components and decomposition time, due to a reduced number of pre-regions for each event. We can see that the second column ("Maximum number of pre-regions for an event") of Table III never exceeds a dozen, therefore the excitation-closure encoding does not cause problems, even if it is exponential.

The decomposition times for FCPNs have still the same order of magnitude of SM decomposition, often the times are smaller because of the usage of a SAT solver during FCPN creation and the merge procedure. There are still some cases which have a higher order of magnitude with respect to others because of a high number of regions extracted from the initial transition system. But it should be noted that FCPNs are not guaranteed to be safe, meanwhile SMs are.

The time spent for the generation of regions is still the same as the one of SMs, therefore it was not reported in the new table and still represents the bottleneck of the decomposition algorithm.

Even if the methods shown previously do not guarantee the safeness of the derived FCPNs, actually only three benchmarks produced unsafe FCPNs: "*vme_write*", "*pparb_2_3*" and "*pparb_2_6*". For these benchmarks Table IV shows the impact of the safeness check, presented in Sec. VIII. This table shows two approaches to guarantee a set of safe FCPNs: one oriented to the performance, actually searching SMs with the method presented in [17], the second approach (size oriented) trying to find only FCPNs at the expense of the decomposition time. Since, searching for safe FCPNs represents the addition of a new constraint, there is a drawback which can affect the number of derived components or the decomposition time. Keeping the decomposition time more or less 50% higher with respect to the case of unsafe FCPNs, the average number of derived components increased from 3 to 12. Searching a solution with a reduced number of components, the decomposition time increases by more or less 13 times, scoring an average of 7.33 components. As expected, we observe an inverse relation between the number of components and the decomposition time. Since cases with unsafe FCPNs are very rare, considering the entire set of available benchmarks, searching a solution with the size oriented approach, on the average the decomposition time is only doubled with respect to unsafe FCPN search, generating on average 2.48 components vs 1.96 unsafe ones.

TABLE IV: Comparison between the best approaches to decompose an LTS into a set of synchronizing FCPNs with and without guarantee on the safeness of the components, in the few cases where unsafe FCPNs can be found.

	Unsafe FCPN search		Safe FCPN search (performance oriented)		Safe FCPN search (size oriented)	
	# comp.	decomp. time [s]	# comp.	decomp. time [s]	# comp.	decomp. time [s]
pparb_2_3	2	0.20	10	0.62	5	19.67
pparb_2_6	4	52.78	16	81.81	10	1092.79
vme_write	3	0.90	10	1.70	7	4.60
AVERAGE	3	17.96	12	28.04	7.33	372.53

X. Conclusion

This paper presents our framework for Transition System decomposition into sets of synchronizing SMs or FCPNs. We showed a novel decomposition method, based on encoding excitation-closure by BDDs, improving already good results for FCPN decomposition. Furthermore, we described different approaches to obtain an FCPN decomposition with only safe components.

We expect further future improvements thanks to the evolution of modern MIS/SAT solvers and BDD encoders.

References

[1] 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.

[2] V. Teren, J. Cortadella, and T. Villa, "Decomposition of transition systems into sets of synchronizing state machines," 2021.

[3] ——, "Decomposition of transition systems into sets of synchronizing free-choice petri nets," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, 2022, pp. 165–173.

[4] ——, "Generation of synchronizing state machines from a transition system: A region-based approach," *International Journal of Applied Mathematics and Computer Science (AMCS)*, vol. 33, no. 1, pp. 133–149, 2023.

[5] W. M. Van der Aalst, "Decomposing process mining problems using passages," in *International Conference on Application and Theory of Petri Nets and Concurrency*. Springer, 2012, pp. 72–91.

[6] ——, "Decomposing Petri nets for process mining: A generic approach," *Distributed and Parallel Databases*, vol. 31, no. 4, pp. 471–507, 2013.

[7] H. Verbeek and W. M. Van der Aalst, "Decomposed process mining: The ILP case," in *International Conference on Business Process Management*. Springer, 2014, pp. 264–276.

[8] D. Taibi and K. Systä, "From monolithic systems to microservices: A decomposition framework based on process mining." in *Proceedings of the 9th International Conference on Cloud Computing and Services Science - CLOSER,*, INSTICC. SciTePress, 2019, pp. 153–164.

[9] 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.

[10] 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.

[11] A. A. Kalenkova, I. A. Lomazova, and W. M. van der Aalst, "Process model discovery: A method based on transition system decomposition," in *International Conference on Applications and Theory of Petri Nets and Concurrency*. Springer, 2014, pp. 71–90.

[12] J. de San Pedro and J. Cortadella, "Mining structured Petri nets for the visualization of process behavior," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016, pp. 839–846.

[13] A. Mokhov, J. Cortadella, and A. de Gennaro, "Process windows," in *2017 17th International Conference on Application of Concurrency to System Design (ACSD)*. IEEE, 2017, pp. 86–95.

[14] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

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

[16] J. Carmona, J. Cortadella, and M. Kishinevsky, "New region-based algorithms for deriving bounded petri nets," *IEEE Transactions on Computers*, vol. 59, no. 3, pp. 371–384, 2009.

[17] 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.

[18] 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.

[19] F. Somenzi, "Cudd: Cu decision diagram package release 2.5. 0," *University of Colorado at Boulder*, 2012.

[20] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev, "Synthesizing Petri nets from state-based models," in *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*. IEEE, 1995, pp. 164–171.

[21] J. Carmona, J.-M. Colom, J. Cortadella, and F. García-Vallés, "Synthesis of asynchronous controllers using integer linear programming," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 25, no. 9, pp. 1637–1651, 2006.

[22] 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.

[23] 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.

[24] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, Pasadena, CA, 2008, pp. 11–15.