Decomposition of transition systems into sets of synchronizing state machines

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*—Transition systems (TS) and Petri nets (PN) are important models of computation ubiquitous in formal methods for modeling systems. An important problem is how to extract from a given TS a PN whose reachability graph is equivalent (with a suitable notion of equivalence) to the original TS.**

This paper addresses the decomposition of transition systems into synchronizing state machines (SMs), which are a class of Petri nets where each transition has one incoming and one outgoing arc and all markings have exactly one token. This is an important case of the general problem of extracting a PN from a TS. The decomposition is based on the theory of regions, and it is shown that a property of regions called excitation-closure is a sufficient condition to guarantee the equivalence between the original TS and a decomposition into SMs.

An efficient algorithm is provided which solves the problem by reducing its critical steps to the maximal independent set problem (to compute a minimal set of irredundant SMs) or to satisfiability (to merge the SMs). We report experimental results that show a good trade-off between quality of results vs. computation time.

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

I. Introduction

The decomposition of a transition system (TS) into a synchronous product of state machines (SMs, Petri nets with exactly one incoming and outgoing edge for every transition) gives an intermediate model between a TS and a Petri net (PN). The set of SMs may exhibit fewer distributed states and transitions, exploiting the best of both worlds of TSs and PNs, leading to better implementations (e.g., smaller circuits with less power consumption). Furthermore, the decomposition procedure extracts explicitly the system concurrency (a PN feature), which is convenient for system analysis and performance improvement (see an example in Fig. 1).

The decomposition of a transition system can be seen from the Petri net perspective as the problem of the coverability by S-components of a Petri net [1]–[3] or of a connected subnet system [4, p. 49] (called S-coverability): each S-component is a strongly connected safe SM i.e., SM with only one token, therefore it cannot contain concurrency. The only concurrency of the system is featured in the interaction of the S-components. In our paper we present how the theory of regions [5] can be used to design a similar procedure starting from a transition system and creating a set of interacting SMs, but without building an equivalent Petri net. Our approach computes a set of minimal regions with the excitation-closure (EC) property of a given TS, and derives from them an irredundant synchronous product of interacting SMs. Excitation closure guarantees that the regions extracted from the transition system are sufficient to model its behaviour.

A. Previous and related work

In [6], a transition system is decomposed iteratively into an interconnection of n component transitions systems with the objective to extract a Petri net from them. This can be seen as a special case of our problem, because in [6] the decomposition allows the extraction of a Petri net, but the decomposed set of transition systems cannot be used as an intermediate model. Their approach is flexible in choosing how to split the original transition system, but it does not provide any minimization algorithm, so that the redundancy due to overlapping states in the component transition systems translates into redundant places of the final Petri net. Another method presented in [7] is based on the decomposition of transition systems into *"slices"*, where each transition system is separately synthesized into a Petri net, and in case of Petri nets "hard" to understand the process can be recursively repeated on one or more *"slices"* creating a higher number of smaller PNs. With respect

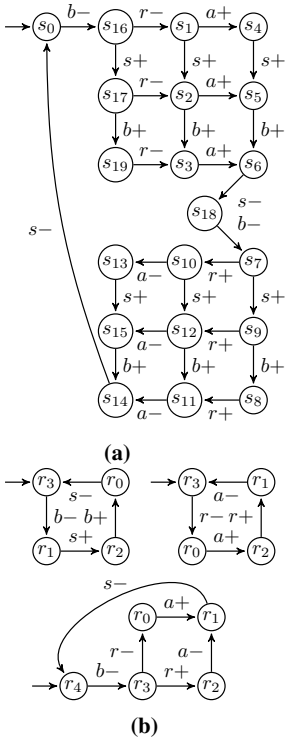


Fig. 1: TS derived from an STG[1](1a) and the derived set of synchronizing state machines (1b).

[1]A Signal Transition Graph (**STG**) $G = (V, E)$ is an interpreted subset of marked graphs wherein each transition represents either the rising (x^+) or falling (x^-) of a signal x which has signal levels high and low. V is the set of transitions and E is the set of edges corresponding to places of the underlying marked graph.

978-1-6654-2703-6/21/$31.00 ©2021 IEEE
DOI 10.1109/DSD53832.2021.00021

to the aforementioned methods, our objective was the creation of a method for the decomposition of TSs in synchronizing SMs without the use of PNs and aiming to the minimization of the computational time and size, applying a minimization criteria.

Decomposition plays an important role in process mining [8]–[11], where in most cases the decomposition starts from a Petri net representing the whole behaviour of the system [8]–[10]. Instead of creating a PN from event logs we can easily create a transition system [12], [13] and directly decompose it with our algorithm.

This paper is organized as follows. Sec. II introduces the background material. Sec. III represents a part of our contribution characterizing and showing a procedure to extract SMs from TSs. Additionally Sec. IV represents the second part of our contribution with the main theoretical result showing that the synchronous product of SMs is bisimilar [5] to the original transition system. Exhaustive experiments are reported in Sec. V, with final conclusions drawn in Sec. VI.

II. Preliminaries

A. Transition systems, Petri nets and synchronizing SMs

We assume the reader to be familiar with Transition systems and Petri nets. We refer to [14], [15] and [5] for a deeper insight on the concepts used 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.

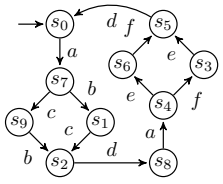


Fig. 2: Example of transition system.

The TS in Fig. 2 will play the role of running example.

We refer to [14] for the definition of synchronous product, adopting the syntax $TS_1||TS_2$ for the synchronous product of TS_1 and TS_2. Being the synchronous product associative, we can define the product of a collection of n TSs: $TS_1||TS_2||\dots||TS_n = ((TS_1||TS_2)\dots)||TS_n$; as an alternative, we can extend directly the definition to more than two TSs.

It has been observed in [4, p. 49] that a state machine $M = (P, T, F, M_0)$ can be interpreted as a transition system $TS = (P, T, \Delta, s_0)$, where the places correspond to the states, the transitions to the events, s_0 corresponds to the unique marked initial place, and $(p, t, p') \in \Delta$ iff ${}^\bullet t = \{p\}$ and $t^\bullet = \{p'\}$ (in a SM by definition $|{}^\bullet t| = |t^\bullet| = 1$). Therefore the reachability graph of M is isomorphic [5] to the transition system *TS*, i.e., *RG*(M) is isomorphic to *TS*.

In this paper we consider sets of synchronizing SMs.

B. Theory of regions

In this paper we propose a procedure for the decomposition of Transition Systems based on the theory of regions (from [5]). 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 some completely inside and some completely outside the region.

For those who is not familiar with the theory of regions we refer to [14] for the definitions of *region*, *minimal region*, *pre-region/post-region*, *excitation set/switching set*, *excitation closure* and *Excitation Closed Transition System (ECTS)*.

III. From LTS to SMs by regions

We now show how to decompose an ECTS into a set of synchronizing SMs. If the initial TS does not satisfy the excitation closure (EC) or event effectiveness property, *label splitting* [5] can be performed to obtain an ECTS.

A set of regions R represents a state machine if R covers all the states S of the transition system and all the regions are disjoint, i.e.:

$$\forall r \in R, \nexists r' \in R : r \cap r' \neq \emptyset \qquad \wedge \qquad \forall s \in S, \exists r \in R : s \in r$$

Given a set of regions satisfying the previous properties we obtain a state machine whose places correspond to the regions, with a transition $r_i \xrightarrow{e} r_j$ when r_i and r_j are pre- and post-regions of e, respectively. Since the regions of an SM are disjoint, each derived SM has only one marked place, which corresponds to the regions that cover the initial state. Notice that *only* the events that cross some region appear in the SM. Notice also that the reachability property of the original TS is inherited by the SMs obtained by this construction.

Theorem 1. *Given an ECTS $TS = (S, E, T, s_0)$ and the set of all its minimal regions, a subset of regions R represents an SM if and only if the set covers all the states of TS and all its regions are pairwise disjoint.*

Proof. The proof can be found in [14]. □

The property of excitation closure can be inherited by the SMs, as stated in the following definition.

Definition 1 (Excitation-closed set of State Machines derived from an ECTS)**.** *Given a set of SMs S derived from an ECTS TS, the set of all regions R of S, the set of labels E of TS, the sets of pre-regions ${}^\circ e$ of the TS for all $e \in E$:*

S is excitation-closed with respect to the regions of TS if the following condition is satisfied:

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

The first step to decompose a transition system is to enumerate all the minimal regions of the original TS. Each collection of disjoint regions covering all the states of the TS represents a state machine, such that the regions are mapped to places of the SM, i.e., each such SM includes a subset of regions of the original TS and represents only the behavior related to the transitions entering into these regions or exiting from them (instead, internal and external events are missing).

The example in Sec. IV shows also that we do not need all the SMs to reconstruct the original LTS, so the question is how many of them we need and which is the "best" (in some sense) subset of SMs sufficient to represent the given LTS. Therefore we may set up a search to obtain a subset of SMs, which are excitation-closed and cover all events, to yield a composition

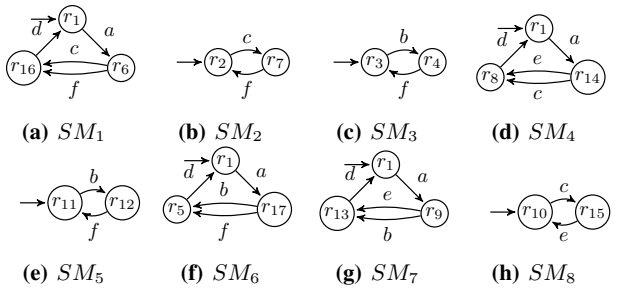


Fig. 3: All SMs created from TS in Fig. 2.

equivalent to the original TS. An easy strategy to guarantee the complete coverage of all events is to add new SMs until all regions are used. However, the resulting collection of SMs may contain completely or partially redundant SMs (see Secs. III-B and III-C), which can be removed exactly or greedily by verifying the excitation-closure property. Moreover, the size of the selected SMs can be reduced through removing redundant labels by merging regions. As a summary, Algorithm 1 shows the decomposition procedure.

Algorithm 1 Decomposition

Require: An ECTS
Ensure: A minimal set of interacting SMs
1: Computation of all minimal regions
2: Generation of a set of SMs with *EC* property
3: Removal of redundant SMs
4: Merge of regions preserving the *EC* property

The first step of the algorithm can be achieved by a greedy algorithm from the literature, which checks minimality while creating regions [5] [4, p. 103] [16].

The second step of the decomposition algorithm 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 which connect intersecting regions. Each *maximal independent set* of the aforementioned graph corresponds to a set of disjoint regions that define an SM.

A greedy algorithm is used for the computation of the third step: starting from the SM with the highest number of regions, one removes each SM whose removal does not invalidate the ECTS properties.

The last step of merging is reduced to a SAT instance, by encoding all the regions of each SM and also the events implied by the presence of one or more regions. Solving this SAT instance by a SAT solver, the number of labels can be minimized by merging the regions which occur multiple times in different SMs.

A. Generation of a set of SMs with excitation closure

Given a set of minimal regions of an excitation-closed TS, Algorithm 2 returns an excitation-closed set of SMs, by associating sets of non-overlapping regions to SMs as mentioned below. Notice that in Def. 1 we extended the definition of an excitation-closed transition system (ECTS) to an excitation-closed set of SMs, by requiring that the two properties of excitation-closure and event-effectiveness hold on the union of regions underlying the SMs.

[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.

Algorithm 2 Generation of excitation-closed set of SMs

Require: Set of minimal regions of an ECTS
Ensure: An excitation-closed set of SMs
1: Create the graph G where each node is a region and there is an edge between intersecting regions
2: $G_0 \leftarrow G$
3: $M \leftarrow \emptyset$, $F \leftarrow \emptyset$
4: **do**
5: Compute $m = MIS(G)$
6: $M \leftarrow M \cup \{m\}$
7: $G \leftarrow G \setminus M$
8: **while** $G \neq \emptyset$
9: **for** $m \in M$ **do**
10: Compute $\tilde{m} = MIS(G_0)$ with the constraint $\tilde{m} \supseteq m$
11: Build state machine $s\tilde{m}$ induced by set of regions $\tilde{m}$
12: $F \leftarrow F \cup \{s\tilde{m}\}$
13: **return** F

Initially, Algorithm 2 converts the minimal regions of the TS into a graph G, where intersecting regions define edges between the nodes of G (line 1). As long as G is not empty, the search of the maximal independent sets is performed on it by invoking the procedure MIS on G (*MIS*(G), line 5), storing the results in M (line 6) and removing the vertices selected at each iteration (line 7). In this way, each vertex will be included in one MIS solution. Notice that the maximal independent sets computed after the first one are not maximal with respect to the original graph G_0, because the MIS procedure is run on a subgraph of G_0 without the previously selected nodes. To be sure that we obtain maximal independent sets with respect to the original G_0, we expand to maximality the independent sets in M, by invoking the MIS procedure on each independent set $m \in M$ constrained to obtain a maximal independent set $\tilde{m} \supset m$ on G_0 (from line 9). Then from the maximal independent sets we obtain the induced state machines to be stored in F (from line 12). The motivation of this step to enlarge the independent sets is to increase the number of regions for each SM, in order to widen the space of solutions for the successive optimizations of redundancy elimination and merging. The set of SMs derived from Algorithm 2 satisfies the EC and event-effectiveness properties because by construction each region is included in at least one independent set.

Fig. 3 shows the resultant SMs derived from the TS in Fig. 2

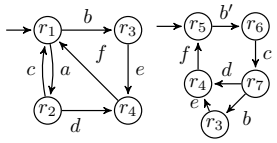
B. Removal of the redundant SMs

The set of SMs generated by Algorithm 2 may be redundant, i.e., it may contain a subset of SMs which still define an ECTS. We describe a greedy search algorithm to obtain an irredundant set of SMs: we order all the SMs by size and try to remove them one by one starting from the largest to the smallest, by checking that the union of the remaining regions satisfies *excitation closure* and *event effectiveness*. If excitation closure and event effectiveness are preserved, then the given SM can be removed. This algorithm is not optimal, because the removal of an SM may prevent the removal of a set of smaller SMs whose sum of places is greater than the number of places

of the removed SM. However, this approach guarantees good performance having linear complexity in the number of SMs.

After the removal of the redundant SMs from the set shown in Fig. 3 only SM_4, SM_5, SM_6 and SM_8 are left.

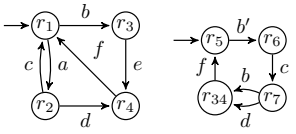
C. Merge between regions preserving the excitation closure



(a) SM_a **(b)** SM_b

Fig. 4: Initial SMs.

The third step of the procedure merges pairs of regions with the objective to minimize the size of the sets of SMs: edges carrying labels are removed, and by consequence the two nodes connected to them are merged decreasing their number. E.g., both the SMs in Fig. 4 (obtained from a TS different from the one in Fig. 2) contain an instance of label e connected by regions r_3 and r_4. This means that an edge carrying label e can be removed in one of the SMs. The result of removing the edge with label e in SM_b and merging the regions r_3 and r_4 replacing them with the region r_{34} is shown in Fig. 5.



(a) SM_a **(b)** SM_b

Fig. 5: SMs of Fig. 4 after the removal of label e in SM_b.

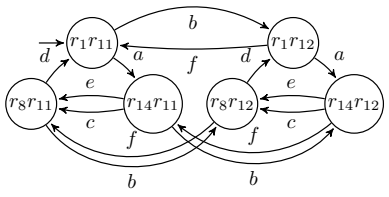
All instances of a region except one can be removed, because removing all of them would change the set of regions used for checking the excitation closure-property, whereas keeping at least one guarantees the preservation of the property.

We formulated the merging problem as solving an instance of SAT. We will skip the exact SAT clause encoding due to lack of space. According to the SAT solution, the SMs are restructured by removing arcs and nodes to be deleted and adding merged nodes, and redirecting arcs as appropriate. In the running example, in SM_b we merge the nodes r_3, r_4 into node r_{34}, remove the edge labeled e between the deleted nodes r_3 and r_4, and redirect to r_{34} the edges pointing to r_3 or r_4.

Instead, none of the four SMs surviving the irredundancy step from Fig. 2 is further minimized by the merging step.

IV. Composition of SMs and equivalence to the original TS

Intuitively, the SMs derived from an LTS interact running in parallel with the same rules of the synchronous product of transition systems. Indeed, if we interpret the reachability graphs of the SMs as LTSs and execute the synchronous product deriving a single LTS which models the interaction of the SMs, it turns out that the result of the composition is equivalent to the original LTS. E.g., consider the composition of reachability graphs of SMs SM_4 and SM_5 in Fig. 6, it generates a superset of behaviors of the original LTS in Fig. 2: it produces the sequence "acbdaefd" which is in the original LTS, but also new behaviors, which are not in the original LTS because some constraints of the original LTS are missing; indeed, these two SMs are not enough to satisfy the excitation-closure property, whereas event effectiveness is satisfied by them because all events are included in the composition. The composition of SMs can exhibit these hidden behaviors by including new regions: e.g., the composition of SM_4 with SM_5 includes two new regions r_{11} and r_{12} so that the events b and f show up in the composition.



(a) $RG(SM_4)||RG(SM_5)$

Fig. 6: Composition between $RG(SM_4)$ and $RG(SM_5)$ of Fig. 3

The equivalence between an ECTS and the derived set of SMs is proved by defining a bisimulation between the original TS and the synchronous product of the reachability graphs of the derived state machines $RG(SM_1)||RG(SM_2)||\ldots||RG(SM_n)$, denoted by $||_{i=1,\ldots,n}RG(SM_i)$.

Theorem 2. *Given an excitation-closed set $\{SM_1, \ldots, SM_n\}$ of SMs derived from the ECTS TS, there is a bisimulation B such that TS $\sim_B ||_{i=1,\ldots,n}RG(SM_i)$.*

Proof. The proof can be found in [14]. □

Theorem 2 states that, given a set of SMs, the excitation closure and event effectiveness of the union of their regions is a necessary and sufficient condition to guarantee that their synchronous product is equivalent to the original TS.

V. Experimental results

We implemented the procedure described in Sec. III and performed experiments on an Intel core running at 2.80GHz with 16GB of RAM. Our software is written in C++ and uses *PBLib* [17] for the resolution of SAT. The resolution of the MIS problem is performed by the *NetworkX* library [18].

Due to lack of space we refer to the tables in our extended version of the paper [14] in which the following information can be found.

The generation of minimal regions is the dominating operation taking more than 60% of the overall time spent; it is exponential in the number of events and with the increase of the input dimensions it becomes the bottleneck shadowing the remaining computations. However it is still possible to decompose quite large transition systems with about 10^6 states and $3 \cdot 10^6$ transitions.

Table I compares the states and transitions of transition systems vs. the places/transitions/crossing arcs of the Petri nets derived by Petrify [16] (columns under PN), and vs. our product of state machines for the first benchmark set. The number of crossing arcs is reported by the *dot* algorithm of *graphviz* [19] and can be considered as a metric of structural simplicity of the model (i.e., fewer crossings implies a simpler structure). Our results from synchronized state machines have similar sizes compared to those from Petri nets, but they have fewer crossings, which is a significant advantage in supporting a visual representation for "large systems". Therefore the plots, in a two-dimensional graphical representation of synchronizing

Example	TS		PN			Synchronizing SMs			
	States	T	P	T	C	SM	P	T	C
alloc-out...	21	18	14	14	3	2	17	21	0
clock	10	10	8	5	4	3	11	15	0
dff	20	24	13	14	21	3	25	41	0
espinalt	27	31	22	20	5	3	29	32	0
fair_arb	13	20	11	10	4	2	12	18	0
future	36	44	18	16	1	3	21	22	0
intel_div3	8	8	7	5	2	2	10	11	0
intel_edge	28	36	11	15	22	4	35	68	1
isend	53	66	25	27	106	13	80	138	4
lin_edac93	20	28	10	8	1	3	13	14	0
master-read	8932	36226	33	26	0	8	38	38	0
pe-rcv-ifc	46	62	23	20	96	2	39	57	2
pulse	12	12	7	6	2	2	7	10	0
rcv-setup	14	17	10	10	5	2	12	14	0
vme_read	255	668	38	29	18	9	50	67	1
vme_write	821	2907	46	33	31	11	57	74	1

TABLE I: Places (P), transitions (T) and arc crossings (C) of the original TS vs. derived Petri nets vs. product of SMs.

SMs, are substantially more *readable* than the ones of Petri nets: see the inputs *intel_edge* and *pe-rcv-ifc* witnessing that peaks of edge crossings are avoided. The example *master-read* instead is an impressive case of how our decomposition tames the state explosion of the original transition system derived from a highly concurrent environment, since from 8932 states we go down to 8 SMs with an average number of 5 states each. An extended version of this table can be found in [14] with details about the resultant SMs and also the comparison with Petri nets computed by an option of Petrify to trade-off more transitions vs. fewer crossings.

We implemented also an exact search of all SMs derived from the original TS, to gauge our heuristics, when it is possible to find an exact solution. We compared the times taken by the exact and heuristic SM generation steps: the exponential behaviour of the exact algorithm makes it hardly affordable for about 15 regions and run out of 16GB of memory for more than 20 regions (the table of comparisons is omitted for lack of space but it can be found in [14]). Instead, the approximate algorithms presented in Sec. III can handle very large transition systems. Even though the result is not guaranteed to be a minimum one, the irredundancy procedure guarantees a form of minimality, yielding a compact representation that avoids state explosion and exhibits concurrency explicitly.

VI. Conclusions

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 transition systems with up to one million states, therefore, it is suitable to handle real cases. 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 MIS and SAT solvers, and the power of HPC since the generation of minimal regions is highly parallelizable. HPC can be exploited also in other steps of the decomposition algorithm, e.g., different MIS computations could be performed simultaneously applying constraints to each parallel computation (e.g., assigning a state to each thread and forcing it to be in the MIS result).

As future work, we want to apply this decomposition paradigm to process mining. Rather than synthesizing intricate "spaghetti" Petri nets from logs, we aim at distilling loosely coupled concurrent threads (SMs) that can be easily visualized, analyzed and optimized individually, while preserving the synchronization with the other threads. Optionally, a new Petri net can be obtained by composing back the optimized threads and imposing some structural constraints, e.g., to be a Free-Choice Petri net, thus providing a tight approximation of the original behavior with a simpler structure.

References

[1] P. Kemper and F. Bause, "An efficient polynomial-time algorithm to decide liveness and boundedness of free-choice nets," in *Application and Theory of Petri Nets*, 1992, pp. 263–278.

[2] J. Desel, *Free choice Petri nets*. Cambridge New York: Cambridge University Press, 1995.

[3] P. M. Mattheakis, "Logic synthesis of concurrent controller specifications," Ph.D. dissertation, University of Thessaly, 2013.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] W. M. Van der Aalst, "Decomposing Petri nets for process mining: A generic approach," *Distributed and Parallel Databases*, vol. 31, no. 4, pp. 471–507, 2013.

[10] 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.

[11] D. Taibi and K. Systä, "From monolithic systems to microservices: A decomposition framework based on process mining." in *CLOSER*, 2019, pp. 153–164.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] NetworkX developer team, "Networkx," 2014. [Online]. Available: https://networkx.github.io/

[19] E. Gansner, E. Koutsofios, S. North, and K.-P. Vo, "A technique for drawing directed graphs," *Software Engineering, IEEE Transactions on*, vol. 19, pp. 214 – 230, 04 1993.