A Region-based Algorithm for Discovering Petri Nets from Event Logs

J. Carmona[1], J. Cortadella[1], and M. Kishinevsky[2]

[1] Universitat Politècnica de Catalunya, Spain
[2] Intel Corporation, USA

Abstract. The paper presents a new method for the synthesis of Petri nets from event logs in the area of Process Mining. The method derives a bounded Petri net that over-approximates the behavior of an event log. The most important property is that it produces a net with the smallest behavior that still contains the behavior of the event log. The methods described in this paper have been implemented in a tool and tested on a set of examples.

1 Introduction

The discovery of formal models from event logs in information systems is known as *process mining*. Since the nineties, the area of process mining has been focused in providing formal support to business information systems [16]. In the industrial domain, ranging from hospitals and banks to sensor networks or CAD for VLSI, process mining can be applied to succinctly summarize the behavior observed in large event logs [14]. Nowadays, several approaches can be used to mine formal models, most of them included in the ProM framework [15].

The *synthesis problem* [7] is related to process mining: it consists in building a Petri net that has a behavior equivalent to a given transition system. The problem was first addressed by Ehrenfeucht and Rozenberg [8] introducing *regions* to model the sets of states that characterize marked places. Process mining differs from synthesis in the knowledge assumption: while in synthesis one assumes a complete description of the system, only a partial description of the system is assumed in process mining. Therefore, bisimulation is no longer a goal to achieve in process mining. Instead, obtaining approximations that succinctly represent the log under consideration are more valuable [19].

In the area of synthesis, some approaches have been studied to take the theory of regions into practice. In [3] polynomial algorithms for the synthesis of bounded nets were presented. This approach has been recently adapted for the problem of process mining in [4]. In [6], the theory of regions was applied for the synthesis of safe Petri nets with bisimilar behavior. Recently, the theory from [6] has been extended to bounded Petri nets [5]. In this paper we adapt the theory from [5] to the problem of process mining.

The work presented in this paper aims at constructing (mining) a Petri net that *covers* the behavior observed in the event log, i.e. traces in the event log

Carmona, J.; Cortadella, J.; Kishinevsky, M. A region-based algorithm for discovering Petri nets from event logs. A: International Conference on Business Process Management. "Business Process Management, 6th International Conference, BPM 2008: Milan, Italy, September 2-4, 2008: proceedings". Springer, 2008, p. 358-373.
The final authenticated version is available online at https://doi.org/10.1007/978-3-540-85758-7_26

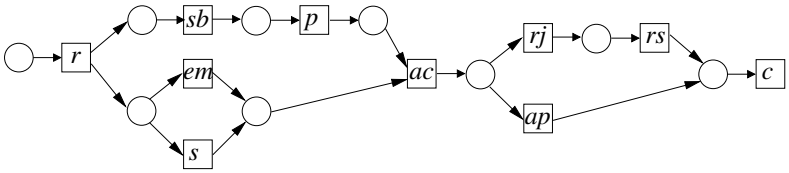

Fig. 1. Petri net mining to avoid overfitting.

will be feasible in the Petri net. Moreover, the Petri net may accept traces not observed in the log. Additionally, a minimality property is demonstrated on the mined Petri net: no other net exists that both covers the log and accepts less traces than the mined Petri net. This capability of minimal over-approximation represents the main theoretical contribution of this paper. The methods presented in the paper can mine a particular k-bounded Petri net, for a given bound k. We have implemented the theory of this paper in a tool, and some preliminary results from logs are reported. The approach taken in this paper is a formal one and differs from the more heuristic methods in the literature. Although the methods presented might have a high complexity for large logs, they can be combined with recent iterative approaches [18] to alleviate their complexity.

This paper shares common goals with the previously presented paper [4]. In [4], two process mining strategies on region of languages are presented, having the same minimality goal as the one that we have in this paper. However the strategy is different: integer linear models are solved in order to find a set of special places called *feasible places* that guarantee the inclusion of the traces from the event log. The more places added, the more traces are forbidden in the resulting net. If the net contains all the possible feasible places, then the minimality property can be demonstrated. However, the set of feasible places might be infinite. In our case, given a maximal bound k for the mining of a k-bounded Petri net, minimal regions of the transition system are enough to demonstrate the minimality property on this bound.

Example. In [14], a small log is presented to motivate the *overfitting* produced by synthesis tools. The log contains the following activities: *r=register*, *s=ship*, *sb=send_bill*, *p=payment*, *ac=accounting*, *ap=approved*, *c=close*, *em=express_mail*, *rj=rejected*, and *rs=resolve*. Now assume that the event log contains the traces *(r, s, sb,p, ac, ap, c)*, *(r,sb,em, p, ac, ap, c)*, *(r, sb, p, em, ac, rj, rs, c)*, *(r, em, sb, p, ac, ap, c)*, *(r, sb, s, p, ac, rj, rs, c)*, *(r, sb, p, s, ac, ap, c)* and *(r, sb, p, em, ac, ap, c)*. From this log, a TS can be obtained [13] and a PN as the one shown in Figure 1 will be synthesized by a tool like `petrify` [6]. If the log is slightly changed (for instance, trace *(r, sb, s, p, ac, rj, rs, c)* is replaced by *(r, sb, s, p, ac, ap, c)*, the synthesis tool will adapt the PN to account for the changes, deriving a different PN. This means that synthesis algorithms are very sensitive to variations in the logs. However, the techniques presented in this paper, as it happens also with traditional min-

ing approaches like the α-algorithm [16], are less sensitive to variations in event logs, and will derive the same PN over the modified log.

The two models used in this paper are *Petri nets* and *transition systems*. We will assume that a transition system represents an event log obtained from observing a real system from which an event-based representation (e.g. a Petri net) approximating its behavior must be obtained. The derivation of the transition system from an event log is an important step, that may have big impact in the final mined Petri net, as it is demonstrated in [13]. A two-step approach is presented in [13], emphasizing that the first step (generation of the transition system) is crucial for the balance between underfitting and overfitting. If the desired abstraction is attained in the first step, i.e. the transition system represents an abstraction of the event log, the second step is expected to reproduce exactly this abstraction, via synthesis. The methods presented in this paper extend the possibilities of this two-step approach, given that the second step might also introduce further abstraction in a controlled manner. The approaches based on regions of languages perform the mining process in only one step, provided that logs can be directly interpreted as languages [4].

2 Preliminaries: theory of regions

2.1 Finite transition systems and Petri nets

Definition 1 (Transition system). *A* transition system *(*TS*) is a tuple* (S, E, A, s_{in})*, where* S *is a set of* states*,* E *is an alphabet of* actions*, such that* $S \cap E = \emptyset$*,* $A \subseteq S \times E \times S$ *is a set of* (labelled) transitions*, and* s_{in} *is the* initial state*.*

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

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

A TS is called *deterministic* if for each state s and each label a there can be at most one state s' such that $s \xrightarrow{a} s'$. The relation between TSs will be studied in this paper. The *language* of a TS, $L(\mathsf{TS})$, is the set of traces feasible from the initial state. When, $L(\mathsf{TS}_1) \subseteq L(\mathsf{TS}_2)$, we will denote TS_2 as an over-approximation of TS_1. The notion of simulation between two TSs is related to this concept:

Definition 2 (Simulation [2]). *Let* $\mathsf{TS}_1 = (S_1, E, A_1, s_{in_1})$ *and* $\mathsf{TS}_2 = (S_2, E, A_2, s_{in_2})$ *be two* TS*s with the same set of events. A* simulation *of* TS_1 *by* TS_2 *is a relation* π *between* S_1 *and* S_2 *such that*

- *for every* $s_1 \in S_1$*, there exists* $s_2 \in S_2$ *such that* $s_1 \pi s_2$*.*
- *for every* $(s_1, e, s_1') \in A_1$ *and for every* $s_2 \in S_2$ *such that* $s_1 \pi s_2$*, there exists* $(s_2, e, s_2') \in A_2$ *such that* $s_1' \pi s_2'$*.*

When TS_1 is simulated by TS_2 with relations π, and viceversa with relation π^{-1}, TS_1 and TS_2 are *bisimilar* [2].

Definition 3 (Petri net [12]). *A Petri net (*PN*) is a tuple* (P, T, F, M_0) *where* P *and* T *represent finite sets of places and transitions, respectively, and* $F \subseteq (P \times T) \cup (T \times P)$ *is the flow relation. The initial marking* $M_0 \subseteq P$ *defines the initial state of the system*[3].

The sets of input and output transitions of place p are denoted by $\bullet p$ and $p\bullet$, respectively. The set of all markings of N reachable from the initial marking m_0 is called its Reachability Set. The *Reachability Graph* of PN ($\mathsf{RG}(\mathsf{PN})$) is a transition system in which the set of states is the Reachability Set, the events are the transitions of the net and a transition (m_1, t, m_2) exists if and only if $m_1 \xrightarrow{t} m_2$. We use $L(\mathsf{PN})$ as a shortcut for $L(\mathsf{RG}(\mathsf{PN}))$.

2.2 Regions

We now review the classical theory of regions for the synthesis of Petri nets [6–8]. Let S' be a subset of the states of a TS, $S' \subseteq S$. If $s \notin S'$ and $s' \in S'$, then we say that transition $s \xrightarrow{a} s'$ *enters* S'. If $s \in S'$ and $s' \notin S'$, then transition $s \xrightarrow{a} s'$ *exits* S'. Otherwise, transition $s \xrightarrow{a} s'$ *does not cross* S'.

Definition 4. *Let* $\mathsf{TS} = (S, E, A, s_{in})$ *be a* TS*. Let* $S' \subseteq S$ *be a subset of states and* $e \in E$ *be an event. The following conditions (in the form of predicates) are defined for* S' *and* e*:*

$$\begin{aligned} \texttt{nocross}(e, S') &\equiv \exists (s_1, e, s_2) \in A : s_1 \in S' \Leftrightarrow s_2 \in S' \\ \texttt{enter}(e, S') &\equiv \exists (s_1, e, s_2) \in A : s_1 \notin S' \wedge s_2 \in S' \\ \texttt{exit}(e, S') &\equiv \exists (s_1, e, s_2) \in A : s_1 \in S' \wedge s_2 \notin S' \end{aligned}$$

The notion of a *region* is central for the synthesis of PNs. Intuitively, each region is a set of states that corresponds to a place in the synthesized PN, so that every state in the region models the marking of the place.

Definition 5 (region). *A set of states* $r \subseteq S$ *in* $\mathsf{TS} = (S, E, A, s_{in})$ *is called a* region *if the following two conditions are satisfied for each event* $e \in E$*:*

- *(i)* $\texttt{enter}(e, r) \;\Rightarrow\; \neg\texttt{nocross}(e, r) \;\wedge\; \neg\texttt{exit}(e, r)$
- *(ii)* $\texttt{exit}(e, r) \;\Rightarrow\; \neg\texttt{nocross}(e, r) \;\wedge\; \neg\texttt{enter}(e, r)$

A region is a subset of states in which *all* transitions labeled with the same event e have exactly the same "entry/exit" relation. This relation will become the predecessor/successor relation in the Petri net. The event may always be either an *enter* event for the region (case (i) in the previous definition), or

[3] Although this paper deals with bounded Petri nets, for the sake of clarity we restrict the theory of current and next sections to the simpler class of safe (1-bounded) Petri nets. Section 4 discusses how to generalize the method for bounded Petri nets.

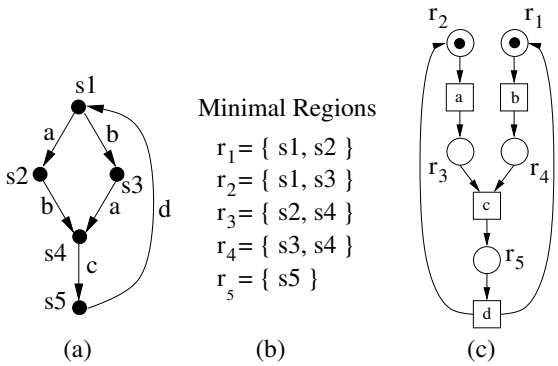

Fig. 2. (a) Transition system, (b) minimal regions, (c) synthesis applying Algorithm of Figure 3.

always be an *exit* event (case (ii)), or never "cross" the region's boundaries, i.e. each transition labeled with e is *internal* or *external* to the region, where the antecedents of neither (i) nor (ii) hold. The transition corresponding to the event will be successor, predecessor or unrelated with the corresponding place respectively.

Examples of regions are reported in Figure 2: from the TS of Figure 2(a), some regions are enumerated in Figure 2(b). For instance, for region r_2, event a is an exit event, event d is an entry event while the rest of events do not cross the region.

Definition 6 (Minimal region). *Let r and r' be regions of a* TS*. A region r' is said to be a* subregion *of r if $r' \subset r$. A region r is a* minimal *region if there is no other region r' which is a subregion of r.*

Going back to the example of Figure 2, in Figure 2(b) we report the set of minimal regions. The union of disjoint regions is a region, so for instance the union of the regions r_1 and r_4 is the set $\{s1, s2, s3, s4\}$ which is also a (non-minimal) region.

Each TS has two *trivial regions*: the set of all states, S, and the empty set. Further on we will always consider only non-trivial regions. The set of non-trivial regions of TS will be denoted by R_{TS}. Given a set $S' \subseteq S$ and a region r, $r\mid_{S'}$ represents the *projection* of the region r into the set S', i.e. $r\mid_{S'} = r \cap S'$.

A region r is a *pre-region* of event e if there is a transition labeled with e which exits r. A region r is a *post-region* of event e if there is a transition labeled with e which enters r. The sets of all pre-regions and post-regions of e are denoted with ${}^\circ e$ and e°, respectively. By definition it follows that if $r \in {}^\circ e$, then all transitions labeled with e exit r. Similarly, if $r \in e^\circ$, then all transitions labeled with e enter r.

Algorithm: PN **synthesis**

- For each event $e \in E$ generate a transition labeled with e in the PN;
- For each minimal region $r_i \in R_{\mathsf{TS}}$ generate a place r_i;
- Place r_i contains a token in the initial marking iff the corresponding region r_i contains the initial state of the TS s_{in};
- The flow relation is as follows: $e \in r_i\bullet$ iff r_i is a pre-region of e and $e \in \bullet r_i$ iff r_i is a post-region of e, i.e.,

$$F_{\mathsf{TS}} \stackrel{def}{=} \{(r,e) | r \in R_{\mathsf{TS}} \ \wedge \ e \in E \ \wedge \ r \in {}^\circ e\} \cup \{(e,r) | r \in R_{\mathsf{TS}} \ \wedge \ e \in E \ \wedge \ r \in e^\circ\}$$

Fig. 3. Algorithm for Petri net synthesis from [11].

2.3 Generation of minimal regions

The computation of the minimal regions is crucial for the synthesis methods in [5,6]. It is based on the notion of *excitation region* [10].

Definition 7 (Excitation region[4]). *The excitation region of an event e, $\mathsf{ER}(e)$, is the set of states in which e is enabled, i.e.*

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

Minimal regions can be generated from the ERs of the events in a TS in the following way: starting from the ER of each event, set expansion is performed on those events that violate the region condition (a pseudocode of the expansion algorithm is given in Figure 10 from [6]). This exploration can be done efficiently by considering only sets that are not supersets of regions already computed [6], because only minimal regions are required. Each time a region is obtained (according to Definition 5), it is added to the set of regions. Finally, from the set of regions computed, non-minimal regions are removed.

2.4 Region-based synthesis

The procedure given by [11] to synthesize a PN, $N_{\mathsf{TS}} = (R_{\mathsf{TS}}, E, F_{\mathsf{TS}}, R_{s_{in}})$, from an *elementary transition system*[5], $\mathsf{TS} = (S, E, A, s_{in})$, is illustrated in Figure 3. Notice that only minimal regions are required in the algorithm [7].

An example of the application of the algorithm is shown in Figure 2. The initial TS and the set of minimal regions is reported in Figures 2(a) and (b), respectively. The synthesized PN is shown in Figure 2(c).

[4] Excitation regions are not regions in the terms of Definition 5. The term is used due to historical reasons.

[5] Elementary transition systems are a proper subclass of the TS considered in this paper, were additional conditions to the ones presented in Section 2.1 are required.

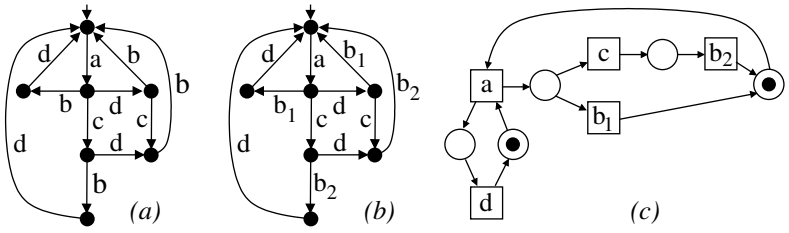

Fig. 4. (a) TS, (b) ECTS by label-splitting, (c) synthesized PN.

2.5 Excitation-closed transition systems

Definition 8 (Excitation-closed TS). *A transition system* $\mathsf{TS} = (S, E, A, s_{in})$ *is called* excitation-closed *(*ECTS*) if it satisfies the following two axioms:*

- *Excitation closure: For each event a:* $\bigcap_{r \in {}^\circ a} r = \mathsf{ER}(a)$
- *Event effectiveness: For each event a:* ${}^\circ a \neq \emptyset$

The synthesis algorithm in Figure 3 applied to an ECTS derives a Petri net with reachability graph *bisimilar* to the initial TS [6]. Note that the *state separation property* of elementary transition systems, which enforces every pair of states to be distinguished by the set of regions is not required. The set of regions needed by the algorithm to preserve bisimilarity can be constrained to minimal pre-regions.

When the TS is not excitation closed, then it must be transformed to enforce that property. One possible strategy is to represent every event by multiple transitions with the same label. This technique is called *label splitting*. Figure 4 illustrates this technique. The initial TS, shown in Figure 4(a) is transformed by splitting the event b into the events b_1 and b_2, as shown in Figure 4(b), resulting in an ECTS. The synthesized PN, with two transitions for event b is shown in Figure 4(c). Hence in PN synthesis label splitting might be crucial for the existence of a PN with bisimilar behavior. However, sometimes label splitting might degrade the resulting PN structure significantly, deriving intricate causality relations that are not helpful for visualization. This phenomenon is discussed in [5].

The following sections introduce PN mining, a version of PN synthesis where the excitation closure is dropped.

3 Algorithm for Petri net mining

The goal of Petri net mining is to generate a PN that over-approximates all observed behaviors in the TS, i.e. $L(\mathsf{TS}) \subseteq L(\mathsf{PN})$, and where $L(\mathsf{PN}) \setminus L(\mathsf{TS})$ is small [4]. Additionally, obtaining a succinct PN with nice causality relations is desirable. For this purpose, the classical synthesis conditions must be adapted

to allow the discovery of behaviors not present in the input TS. In this section we show a simple yet powerful approach for relaxing the region-based synthesis conditions from [5,6] to obtain over-approximations of the TS. Formally, given a TS=(S, E, A, s_{in}) , the theory of regions can be adapted for mining a Petri net PN=(P, T, F, M_0) with the following characteristics:

1. $L(\mathsf{TS}) \subseteq L(\mathsf{PN})$,
2. $T = E$, i.e. there is no label splitting, and
3. *Minimal language containment* (MLC) property:

$$\forall \mathsf{PN}' = (P', T', F', M_0') \; s.t. \; T' = E : L(\mathsf{TS}) \subseteq L(\mathsf{PN}') \Rightarrow L(\mathsf{PN}) \subseteq L(\mathsf{PN}')$$

Therefore the obtained Petri net represents the minimal over-approximation of the input TS that can be synthesized without label splitting. The remainder of this section will show how to relax the region-based synthesis to derive such Petri net.

3.1 Mining over-approximations of a TS

Bisimilarity or language equivalence are very restricting equivalence relations, not very useful for the area of Petri net mining where over-approximations of the initial event log are more valuable [4,19]. In [5,6], bisimulation between the TS and the synthesized PN holds due to the excitation closure condition. Let us assume in this section that the excitation closure condition is dropped, i.e. the set of minimal pre-regions of some events may properly include the excitation region of the event. With this simple relaxation, the PN obtained by the Algorithm of Figure 3 will satisfy the MLC property (Theorem 2).

Theorem 1. *Let* TS=(S, E, A, s_{in}) *be a transition system, and* PN=(P, T, F, M_0) *be the synthesized net with the set of minimal regions of* TS*, using Algorithm of Figure 3. Then* $L(\mathsf{TS}) \subseteq L(\mathsf{PN})$.

Proof. The proof corresponds to the sufficiency direction from Theorem 3.4 in [6]. The theorem guarantees bisimilarity between an ECTS and the reachability graph of the synthesized Petri net from the set of minimal regions. From the two simulations necessary for having bisimulation in that theorem, only one is preserved if the excitation closure condition is dropped. This remaining simulation is the one between the TS and the reachability graph of the PN. Hence every trace in the TS can be simulated by the PN when minimal regions are used, even if the TS is not excitation closed. This suffices to prove the theorem. □

Moreover, as the following results show, regions are preserved under language containment or simulation.

Lemma 1. *Let* TS=(S, E, A, s_{in}), TS$' = (S', E', A', s_{in})$ *be two transition systems such that* $S \subseteq S'$, $E \subseteq E'$, $T \subseteq T'$. *If* $r \in R_{\mathsf{TS}'}$ *then* $r \mid_S \in R_{\mathsf{TS}}$.

Proof. If predicates *(i),(ii)* in Definition 5 hold in r for transitions in A', then they also hold for the transitions in A when r is restricted to S, given that $A \subseteq A'$ and $S \subseteq S'$, i.e. by removing arcs, no new violations of the region conditions can be created (see Definition 5). □

We now prove a similar lemma on the correspondence of regions between simulated TSs.

Lemma 2. *Let* $\mathsf{TS}=(S, E, A, s_{in})$, $\mathsf{TS}' = (S', E, A', s'_{in})$ *be such that there exists a simulation relation of* TS *by* TS′ *with relation* π. *If* $r \in R_{\mathsf{TS}'}$, *then* $\pi^{-1}(r) \in R_{\mathsf{TS}}$, *and the* `nocross/enter/exit` *predicates for every event at* r *are preserved in* $\pi^{-1}(r)$.

Proof. The proof for this lemma is similar to the one used in Lemma 1, but on simulated states: for every transition $(s, e, s') \in A$ there exists a transition $(\pi(s), e, \pi(s')) \in A'$. Therefore, the predicates *(i),(ii)* in Definition 5 hold in TS for the set $\pi^{-1}(r)$. □

In general, language containment between two TSs does not imply simulation [9]. However, if the TS corresponding to the superset language is deterministic then language containment guarantees the existence of a simulation:

Lemma 3. *Let* $\mathsf{TS}_1 = (S_1, E_1, A_1, s_{in1})$ *and* $\mathsf{TS}_2 = (S_2, E_2, A_2, s_{in2})$ *be two* TSs *such that* TS_2 *is deterministic, and* $L(\mathsf{TS}_1) \subseteq L(\mathsf{TS}_2)$. *Then* TS_2 *is a simulation of* TS_1.

Proof. The relation $\pi \subseteq (S_1 \times S_2)$ defined as follows:

$$s_1 \pi s_2 \Leftrightarrow \exists\ \sigma : s_{in1} \xrightarrow{\sigma} s_1 \wedge s_{in2} \xrightarrow{\sigma} s_2$$

represents a simulation of TS_1 by TS_2: the first item of Definition 2 holds since $L(\mathsf{TS}_1) \subseteq L(\mathsf{TS}_2)$. If the contrary is assumed, i.e. $\exists s_1 \in S_1 : \nexists s_2 \in S_2 : s_1 \pi s_2$ then the trace leading to s_1 in TS_1 is not feasible in TS_2, which contradicts the assumption. The second item holds because the first item and the determinism of TS_2: for every $s_1 \in S_1$, TS_2 deterministic implies that there is only one state possible $s_2 \in S_2$ such that $s_1 \pi s_2$. But now if e is enabled in s_1 and not enabled in s_2 will imply that the trace σe, with $s_{in1} \xrightarrow{\sigma} s_1$, is not feasible in TS_2, reaching a contradiction to $L(\mathsf{TS}_1) \subseteq L(\mathsf{TS}_2)$. □

And now we can proof the MLC property on the mined Petri net from a TS:

Theorem 2. *Let* $\mathsf{PN}=(P, T, F, M_0)$ *be the synthesized net with the set of minimal regions of* $\mathsf{TS}=(S, E, A, s_{in})$, *using Algorithm of Figure 3. Then* PN *satisfies the* MLC *property.*

Proof. By contradiction. Let $\mathsf{TS}' = (S', E', A', s'_{in})$ be the reachability graph corresponding to a $\mathsf{PN}' = (P', T', F', M'_0)$ such that $E' = T$, $L(\mathsf{TS}) \subseteq L(\mathsf{TS}')$ and $L(\mathsf{PN}) \not\subseteq L(\mathsf{TS}')$. The following facts can be observed:

- TS' and $RG(\mathsf{PN})$ are deterministic because $E = E' = T$ and therefore they correspond to the reachability graph of Petri nets with a different label for each transition.
- Since TS' is deterministic and $L(\mathsf{TS}) \subseteq L(\mathsf{TS}')$, then there is a simulation π of TS by TS' (Lemma 3).
- $\forall r' \in R_{\mathsf{TS}'}$, $r = \pi^{-1}(r) \in R_{\mathsf{TS}}$, and the `nocross`/`enter`/`exit` predicates of the events is the same in r' and r (Lemma 2).
- Each non-minimal region can be described as the union of disjoint minimal regions [6], and therefore we can focus only on minimal regions.
- Each minimal region $r \in R_{\mathsf{TS}}$ is a region in $R_{RG(\mathsf{PN})}$, since PN has been obtained with Algorithm of Figure 3 from the set of minimal regions in TS. Moreover, since $RG(\mathsf{PN})$ is deterministic and $L(\mathsf{TS}) \subseteq L(\mathsf{PN})$ (Theorem 1), then there is a simulation of TS by $RG(\mathsf{PN})$ (Lemma 3). Now using Lemma 2, together with the fact that r is a region both in R_{TS} and $R_{RG(\mathsf{PN})}$, the `nocross`/`enter`/`exit` predicates of events in TS hold also in $RG(\mathsf{PN})$.
- Hence, the previous items show that for a region in TS' there is a corresponding region in $RG(\mathsf{PN})$ with the same `nocross`/`enter`/`exit` predicates on the events. In Petri net terms, this fact means that the flow relation of PN' is included in the flow relation of PN. Additionally, the simulations connecting both transition systems preserve the initial states (see Lemma 3). This contradicts the assumption that $L(\mathsf{PN}) \not\subseteq L(\mathsf{TS}')$. □

3.2 Related issues and further extensions

Now we discuss the features of the mining strategy and possible extensions.

Visualization capabilities. By removing the excitation closure condition, one guarantees that there is a 1-to-1 correspondence between the events in the log and the transitions in the Petri net. This is important in terms of visualization. Moreover, the set of minimal regions can include *redundant* regions: a region r is redundant if the language of the Petri net without r is preserved. Therefore redundant regions can be safely removed from the net. The theory in [5, 6] proposes methods to detect redundant places, based in the preservation of the excitation closure. Those methods have been adapted and included in the mining approach presented in this paper.

Mining of Petri net subclasses. As it has been done in synthesis (see [6], Section 4.4), the approach presented in this paper might be adapted to mine Petri net subclasses. The basic idea is to restrict the generation of regions in order to generate regions satisfying structural Petri net conditions. Let us use the example in Figure 5 to illustrate this. In Figure 5(b) we report the mined two-bounded Petri net from the TS of Figure 5(a) (next Section shows how to generalize the mining method to the bounded case). Now imagine that we are interested in the mining of *marked graphs*, i.e. Petri nets where places have at most one predecessor (successor) transition. Notice that place p in Figure 5(b) does not satisfy this condition. If the mining of a marked graph is applied, the

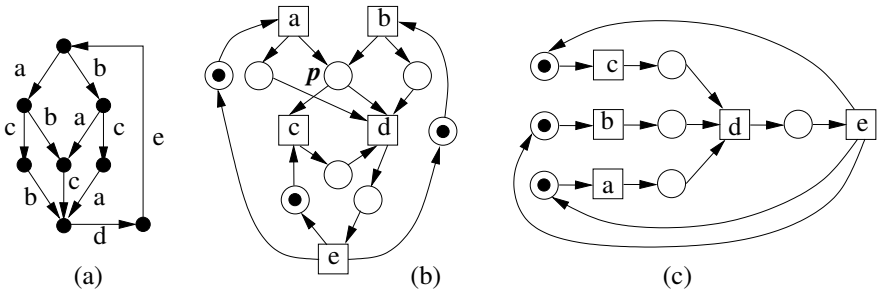

Fig. 5. (a) Initial TS, (b) Mined Petri net , (c) Mined marked graph.

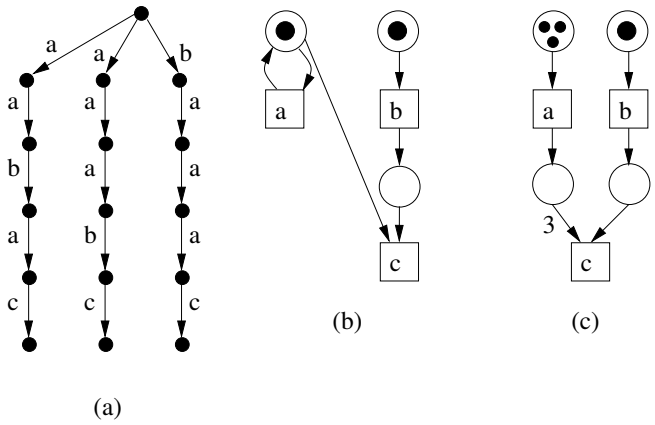

Fig. 6. (a) Transition system, (b) Mined safe Petri net, (c) Mined 3-bounded Petri net.

Petri net shown in Figure 5(c) is obtained.

Critical events. The methods presented can be extended to select those events that might be *critical* in terms of representation: for those events, the avoidance of over-approximation might be imposed by requiring excitation closure on them. The application of label-splitting can be guided to attain this goal.

4 Mining bounded Petri nets

In the literature for the mining of Petri nets from event logs, it is widely accepted the use of ordinary and safe Petri nets for the discovery of process models (a remarkable exception is presented in [4]). Due to the recent results for the synthesis of bounded and weighted Petri nets [5], this limitation can be waved, and therefore a more succinct and accurate model of the log can be obtained using the techniques developed in this paper. Moreover, the possibility to search for unsafe regions might be crucial in order not to over-approximate the event log

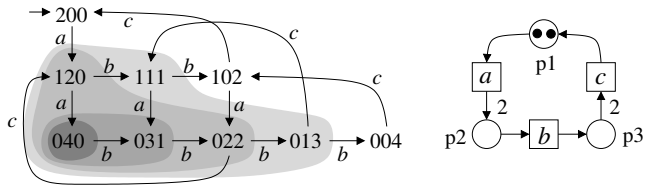

Fig. 7. A transition system and an equivalent bounded Petri net.

too much. To illustrate this fact, see the example in Figure 6. The mining of a safe Petri net from the TS of Figure 6(a) is shown in Figure 6(b), whereas Figure 6(c) reports the mining of a 3-bounded net. The language accepted by the PN from Figure 6(b) is $(a* \parallel b)c$ which might be an over-conservative approximation[6], while the net in Figure 6(c) accepts $(a^3 \parallel b)c$, which although being also an over-approximation, it is a more accurate one. This section introduces informally how the theory of the previous sections can be generalized to mine bounded systems.

In [5], an extension of the region-based synthesis of Petri nets has been presented to support bounded nets. The methods assume that a k is initially given for the search of a k-bounded Petri net. Let us use the example in Figure 7 to summarize the theory. In the bounded case, the basic idea is that regions are represented by multisets (i.e., a state might have multiplicity greater than one). Figure 7 depicts a TS with 9 states and 3 events. After synthesis, the Petri net at the right is obtained. Each state has a 3-digit label that corresponds to the marking of places p_1, p_2 and p_3 of the Petri net, respectively. The shadowed states represent the general region that characterizes place p_2. Each grey tone represents a different multiplicity of the state (4 for the darkest and 1 for the lightest). Each event has a gradient with respect to the region (+2 for a, -1 for b and 0 for c). The gradient indicates how the event changes the multiplicity of the state after firing. For the same example, the equivalent *safe* Petri net has 5 places and 10 transitions.

In summary, the generalization of the theory of Sections 2 and 3 is based on the idea that regions are no longer sets but multisets, and the predicates for region conditions must take into account the gradient of each event on the multisets. The excitation closure notion is defined on the support (states with multiplicity greater or equal than one) of the multiset. Finally, the algorithm for synthesis of bounded Petri nets is generalized to account for bounded markings and weighted arcs. The interested reader may refer to [5] for details.

The theory in [5] has been incorporated in the Petri net mining approach presented in this paper. Hence the mining of Petri nets can be guided to find bounded Petri nets. An example of k-bounded mining is shown in Section 5.

[6] The expression $e_1 \parallel e_2$ denotes the set of possible interleavings between e_1 and e_2.

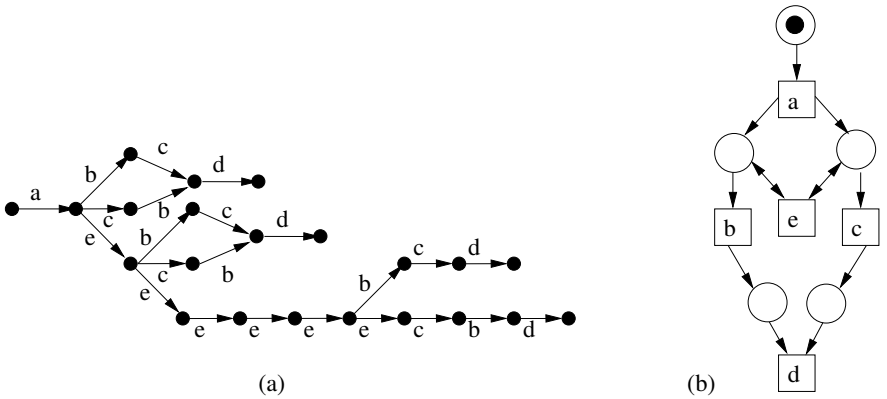

Fig. 8. (a) Initial TS, (b) Mined Petri net.

5 Examples, experiments and tool

The theory described in this paper has been incorporated in `Genet`, a tool for the synthesis and mining of concurrent systems [5]. Most of the examples have been obtained from [1].

Mining of safe Petri nets

Some examples have been presented along the paper. An additional example is shown in Figure 8. The language accepted by the PN of Figure 8(b) is $ae*(b \parallel c)d$, which properly includes the language of the TS of Figure 8(a). Remarkably, applying the α-algorithm [17] to this event log results in the same PN.

Mining of bounded Petri nets

An example of the power of k-bounded PN mining is shown in Figure 9. The system modeled represents 5 procesess sharing 3 resources. Every process requires one resource, but there is one process that requires two resources. We assume that the TS used for this example can be constructed from a set of simulations. The 3-bounded PN from the corresponding transition system contains 20 states and 74 arcs. The synthesis of a safe PN from the transition system applies many label splittings in order to enforce the excitation closure, deriving in a PN with 15 places, 34 transitions and 128 arcs. Clearly, neither the initial TS nor the synthesized PN are of any help to realize the control flow of this example. However, the mined 3-bounded PN is a succint representation of the log.

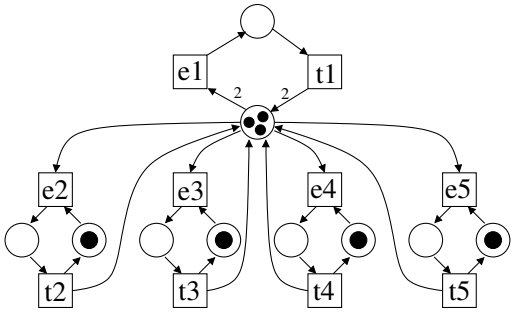

Fig. 9. Mined 3-bounded PN for a system of five processes sharing three resources.

				synthesis		mining									
				petrify		Genet		Genet		ProM			ProM		
				safe		safe		2-bounded		Parikh			Heuristics		
benchmark	$\|S\|$	$\|[S]\|$	$\|E\|$	$\|P\|$	$\|T\|$	$\|P\|$	$\|[S]\|$	$\|P\|$	$\|[S]\|$	$\|P\|$	$\|T_U\|$	$\|[S]\|$	$\|P\|$	$\|T_I\|$	$\|[S]\|$
groupedFollowsa7	18	10	7	7	7	6	11	7	11	7	0	10	7	1	8
groupedFollowsal1	15	15	7	8	9	10	16	12	15	7	0	7	14	10	22
groupedFollowsal2	25	25	11	15	11	15	25	15	25	11	0	13	15	3	25
herbstFig6p21	16	16	7	11	13	7	22	11	16	1	6	2	18	15	∞
herbstFig6p34	32	32	12	16	13	16	34	18	32	8	2	12	19	12	∞
herbstFig6p41	20	18	14	16	14	16	18	16	18	17	0	18	14	0	18
staffware_15	31	24	19	20	20	18	22	19	31	18	0	21	19	0	19
pn_ex_10	233	210	11	64	218	13	281	16	145	8	2	14	41	25	∞

Table 1. PN mining applied to event logs from [1].

Experiments

The mining of some examples is summarized in Table 1. Following the two-step mining approach from [13], we have obtained the transition systems from each log with the *FSM Miner* plugin available in ProM. For each log, columns report the number of states of the initial log $|S|$, number of states of the minimal bisimilar transition system $|[S]|$ (that gives an idea of the amount of redundancy present in the initial log) and number of events $|E|$. Next, the number of places $|P|$ and transitions $|T|$ of the PN obtained by synthesis is reported. For each version of the mining algorithm (safe and 2-bounded), the number of places of the mined PN and number of states of the corresponding minimal bisimilar reachability graph are reported. The CPU time for the mining of all examples but the last one has taken less than two seconds. The mining of pn_ex_10, 2-bounded version, took one minute. Finally the same information is provided for two well-known mining algorithms in ProM: the *Parikh Language-based Region* and the *Heuristics* [20] miners. The number of unconnected transitions ($|T_U|$)

derived by the Parikh miner and the number of invisible transitions introduced by the Heuristic miner is also reported ($|T_I|$).

The numbers in Table 1 suggest some remarks. If the synthesis is compared with the mining in the case of safe PNs, it should be noticed that even for those small examples the number of transitions is reduced, due to the absence of label splitting (see row for pn_ex_10). The number of places is also reduced in general. It should also be noticed that 2-bounded mining represents the log more accurately, and thus more places are needed with respect to the mining of safe nets. Sometimes the mined PN accepts more traces but the corresponding minimal bisimilar transition system has less states, e.g. pn_ex_10: after over-approximating the initial TS, several states become bisimilar and can be minimized.

The Parikh miner tends to derive very aggressive abstractions, as it is demonstrated in the pn_ex_10 and herbstFig6p21 logs. Sometimes the Petri nets obtained with this miner contain isolated transitions, because the miner could not find places connecting them to the net. The Heuristics miner is based on the frequency of patterns in the log. The miner derives a heuristic net that can be afterwards converted to Petri net with ProM. Some of the Petri nets obtained with this conversion turned out to be unbounded (denoted with symbol ∞ in the table), and contain a significant amount of invisible transitions. This miner is however very robust to noise in the log. In conclusion, different miners can achieve different mining goals, widening the application of Process mining into several directions.

6 Conclusions

A strategy for adapting the theory of regions for the area of Process mining has been presented. The main contribution is to allow the generation of over-approximations of the event log by means of a bounded Petri net, not necessarily safe. An important result is presented that guarantees the minimal language containment property on the mined PN. The theory has been incorporated in a synthesis tool.

Acknowledgements

We would like to thank W. van der Aalst, E. Verbeek and B. van Dongen for helpful discussions and guidance, and anonymous referees for their help in improving the final version of the paper. This work has been supported by the project FORMALISM (TIN2007-66523), and a grant by Intel Corporation.

References

1. Process mining. www.processmining.org.
2. A. Arnold. *Finite Transition Systems*. Prentice Hall, 1994.

3. E. Badouel, L. Bernardinello, and P. Darondeau. Polynomial algorithms for the synthesis of bounded nets. *Lecture Notes in Computer Science*, 915:364–383, 1995.
4. R. Bergenthum, J. Desel, R. Lorenz, and S.Mauser. Process mining based on regions of languages. In *Proc. 5th Int. Conf. on Business Process Management*, pages 375–383, Sept. 2007.
5. J. Carmona, J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. A symbolic algorithm for the synthesis of bounded Petri nets. In *29th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency*, June 2008.
6. J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri nets from finite transition systems. *IEEE Transactions on Computers*, 47(8):859–882, Aug. 1998.
7. J. Desel and W. Reisig. The synthesis problem of Petri nets. *Acta Inf.*, 33(4):297–315, 1996.
8. A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures. Part I, II. *Acta Informatica*, 27:315–368, 1990.
9. J. Engelfriet. Determinacy - (observation equivalence = trace equivalence). *Theor. Comput. Sci.*, 36:21–25, 1985.
10. M. Kishinevsky, A. Kondratyev, A. Taubin, and V. Varshavsky. *Concurrent Hardware: The Theory and Practice of Self-Timed Design*. John Wiley and Sons, London, 1993.
11. M. Nielsen, G. Rozenberg, and P. Thiagarajan. Elementary transition systems. *Theoretical Computer Science*, 96:3–33, 1992.
12. C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Bonn, Institut für Instrumentelle Mathematik, 1962. (technical report Schriften des IIM Nr. 3).
13. W. van der Aalst, V. Rubin, H. Verbeek, B. van Dongen, E. Kindler, and C. Günther. Process mining: A two-step approach to balance between underfitting and overfitting. Technical Report BPM-08-01, BPM Center, 2008.
14. W. M. P. van der Aalst and C. W. Günther. Finding structure in unstructured processes: The case for process mining. In T. Basten, G. Juhás, and S. K. Shukla, editors, *ACSD*, pages 3–12. IEEE Computer Society, 2007.
15. W. M. P. van der Aalst, B. F. van Dongen, C. W. Günther, R. S. Mans, A. K. A. de Medeiros, A. Rozinat, V. Rubin, M. Song, H. M. W. E. Verbeek, and A. J. M. M. Weijters. ProM 4.0: Comprehensive support for *eal* process analysis. In J. Kleijn and A. Yakovlev, editors, *ICATPN*, volume 4546 of *Lecture Notes in Computer Science*, pages 484–494. Springer, 2007.
16. W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, 2003.
17. W. M. P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.
18. B. van Dongen, N. Busi, G. Pinna, and W. van der Aalst. An iterative algorithm for applying the theory of regions in process mining. Technical Report Beta rapport 195, Department of Technology Management, Eindhoven University of Technology, 2007.
19. H. Verbeek, A. Pretorius, W.M.P. van der Aalst, and J.J. van Wijk. On Petri-net synthesis and attribute-based visualization. In *Proc. Workshop on Petri Nets and Software Engineering (PNSE'07)*, pages 127–141, June 2007.
20. A. Weijters, W. van der Aalst, and A. A. de Medeiros. Process mining with the heuristics miner-algorithm. Technical Report WP 166, BETA Working Paper Series, Eindhoven University of Technology, 2006.