Synthesis of Elementary Net Systems with Final Configurations

Benjamin Meis, Robin Bergenthum, Jörg Desel

Department of Software Engineering,
FernUniversität in Hagen
{firstname.lastname}@fernuni-hagen.de

Abstract. In this paper, we extend the definition of transition systems, as well as the definition of elementary net systems, by a notion of final states and provide a solution to the synthesis problem of elementary net systems with final configurations. Furthermore, we present how to simplify complex sets of final configurations of an elementary net system by introducing the notion of neat places. A place is neat if it is marked only if the elementary net system is in a final configuration. If it is not possible to generate a neat place, we present a heuristic to identify so-called neat terms indicating final configurations.

1 Introduction

Petri nets are a well established model for concurrent and distributed systems. They have a formal semantics as well as an intuitive graphical representation. The most basic class of Petri nets is given by elementary net systems. Fundamental concepts such as causality and choice are well defined for elementary net systems. Therefore, this class is the formal basis for many modern business process modeling languages, like Event Driven Process Chains (EPC) [11], Business Process Modeling Notation (BPMN) [10] or activity diagrams in the UML standard [12].

The sequential behavior of an elementary net system is modeled by means of its reachability graph, which is a transition system, modeling the set of reachable states as well as the state transitions of the elementary net system. Although the reachability graph of an elementary net system can be exponential in the size of the elementary net system, it can easily be defined and constructed. Conversely, assuming a transition system modeling the behavior of a system in terms of transitions, the synthesis problem is to generate an elementary net system so that its reachability graph is isomorphic to the transition system, i.e., both graphs with labeled arcs are identical up to the names of states. Ehrenfeucht and Rozenberg presented a solution of the synthesis problem in the early 90s, introducing the so-called theory of regions [5, 6]. The recently published textbook [4] provides a good survey to this topic.

In this paper, we tackle the synthesis problem extending the behavioral model, i.e. the transition system, as well as the elementary net system by notions of final states. In many application areas, like business processes and protocols, behavior is constituted by a set of runs which lead to a final state each. Modeling final states on top of the behavior of a system increases the accuracy of the model. Therefore, the set of states

of a real world system consists of two disjunct parts: a set of final states and a set of intermediate states. If a system is in a final state, the current execution of this system is considered to be complete. If the current execution of a system ends at an intermediate state, the occurred behavior is invalid.

We model the behavior of a system with final states extending the transition system by a set of final states. Transition systems with final states are closely related to finite state automata (see e.g. [7]). A finite state automaton recognizes words; the set of all words recognized is called the language of the automaton. Equivalently, the automaton can be considered to generate the words of its language. A word, i.e. a sequence of symbols, is recognized (or generated), if the automaton, by reading the symbols subsequently, moves from state to state and ends in a final state. Identifying symbols of automata with activities modeled in transition systems, the language of a transition system with end states is the language of the related automaton.

Since finite state automata have final states, their behavior is not considered to be prefix-closed. In other words, even if a word is not accepted, it can be the prefix of another word, which belongs to the language of the automaton. Translated to transition systems, we can consider a system's behavior illegal even if it can be extended to a legal one; so a legal run is considered to be a complete run.

Regarding Petri nets, it is usually assumed that every prefix of a valid execution is also valid. This comes with the formal definition of firing sequences and both behavioral notions. However, Petri net classes tailored for particular application areas such as workflow nets [1] and C-nets [3], do consider a single, distinguished final state. In this sense, these classes are a special case of the class of Petri nets we consider in this paper. In our more general setting, every reachable state of a Petri net is a possible final state. In an elementary net system, a state is a configuration of the net. Thus, we consider elementary net systems with final configurations.

In this paper, we provide a solution to the synthesis problem of elementary net systems with final configurations. This problem is formally stated as follows: let there be a transition system with final states, generate an elementary net system with a set of final configurations so that the reachability graph of this net is isomorphic to the transition system, and the respective sets of final states match. It will turn out that this solution is a straightforward generalization of the solution of elementary net systems without final states; every solution to that problem can be straightforwardly extended by final states. As an example, we consider the transition system depicted in Figure 1a. Figure 1b depicts a possible solution of the related synthesis problem. Figure 3 depicts the same transition system extended by a set of final states. Figure 2 depicts a possible solution of the related synthesis problem with final states.

Although the set of final states of a transition system can be quite arbitrary, the set of final states of the synthesized net should have some particular structure. For example, for workflow nets a configuration is final if and only if it contains only a distinguished final place. We explore a similar concept and consider sets of final configurations of elementary net systems which can be identified checking a single place; a configuration is final if and only if it contains this place. We call places with this property neat places.

Considering a transition system with final states, we consider the problem to synthesize a corresponding net with a neat place, if this is possible. We characterize potential

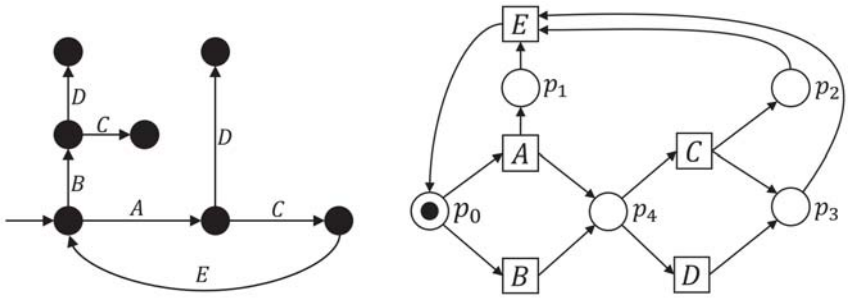


Fig. 1. Synthesizing an elementary net system from a transition system

neat places in terms of regions of transition systems. We show that, if a neat place exists, it either belongs already to a synthesized net, or the net can be extended by a neat place.

Neat places have the particular advantage of representing a set of final states in a very efficient and easily identifiable way. Actually, a neat place can represent an exponential number of reachable configurations. Unfortunately, neat places do not always exist. We will finally show in this paper that also for cases where there are no neat places we can often represent a big set of final configurations in an efficient way. To this end, we define terms of places, constructed by means of place names and logical connectors, so that a term is evaluated to true in a given configuration if and only if this configuration is final. We show how to obtain these terms in a systematic way.

The remainder of the paper is organized as follows. In Section 2 we present the definitions on elementary net systems and transition systems, extend both definitions by final configurations and final states respectively. We decide the synthesis problem for elementary net systems with final configurations. In Section 3 we introduce the notion of a neat place and in Section 4 we introduce the notion of a neat term to indicate final configurations. Section 5 concludes the paper and presents an outlook of further work.

2 Synthesis of Elementary Net Systems with Final States

In this section we will recapitulate the notions of elementary net systems and transition systems [9] and extend elementary net systems by final configurations and transition systems by final states. At the end of this section we show how to solve the corresponding synthesis problem.

We consider elementary net systems with final configurations in this paper. As stated in the introduction, we use the notion of final configurations to define a set of states of an elementary net in which we consider its behavior to be complete.

Definition 1 (Elementary Net System). *An elementary net system is a tuple* $N = (P, T, A, m_0)$ *where* P *is a finite set of places,* T *is a finite set of transitions, satisfying* $P \cap T = \emptyset$*,* $A \subseteq (P \times T) \cup (T \times P)$ *is a set of directed arcs, and* $m_0 \subseteq P$ *is the initial configuration.*

The tuple (N, C) *where* $C \subseteq \{m | m \subseteq P\}$ *is an elementary net system with a set of final configurations* C*.*

Elementary net systems have a simple firing rule. We call $\bullet t = \{p \in P \mid (p,t) \in A\}$ the preset of a transition t and $t\bullet = \{p \in P \mid (t,p) \in A\}$ the postset of t. A transition $t \in T$ is enabled in a configuration m, if $\bullet t \subseteq m$ and $t\bullet \cap m = \emptyset$. If t is enabled in a configuration m, t can fire changing the configuration m to m' defined by $m' = (m \setminus \bullet t) \cup t\bullet$. We write $m \xrightarrow{t} m'$, if m enables t and the occurrence of t changes m to m'.

Let $\sigma = t_1 \dots t_n$ be a sequence of transitions. We write $m \xrightarrow{\sigma} m'$, if there are configurations $m_1, m_2, \dots, m_{n-1}$ so that $m \xrightarrow{t_1} m_1 \xrightarrow{t_2} m_2 \dots m_{n-1} \xrightarrow{t_n} m'$ holds.

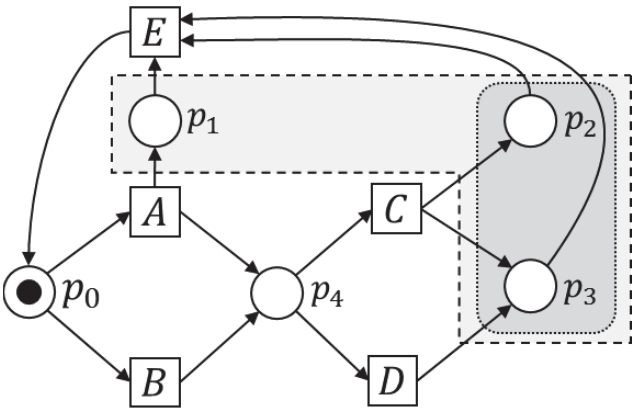


Fig. 2. An elementary net system with two final configurations.

Figure 2 depicts our running example net with final configurations in its initial configuration. The net has five transitions A, B, C, D, E and five places. The initial configuration is the set $\{p_0\}$. Graphically, a configuration is depicted by tokens marking all places of the configuration. In the initial configuration both transitions A and B are enabled. Firing A removes p_0 and adds the places p_1 and p_4 to the configuration of the net. The two final configurations of this example, $\{p_2, p_3\}$ and $\{p_1, p_2, p_3\}$, are depicted by dashed frames. In general, every set of places may be defined as a final configuration. We reach the first final configuration by firing the sequence B, C. Another possibility is firing A, C, E, B, C. One possibility to reach the second final configuration is firing A, C. The final configuration $\{p_2, p_3\}$ is a so-called deadlock, i.e. a configuration where no transition is enabled. The final configuration $\{p_1, p_2, p_3\}$ enables transition E.

The behavior of an elementary net system can be expressed by terms of a transition system. Every configuration of the net constitutes a node, called state, and every enabled transition of the net is a state transition of the transition system. A final configuration of the elementary net system relates to a final state of the transition system.

Definition 2 (Transition System). *A transition system is a tuple $TS = (S, T, \Theta, s_0)$, where S is a finite set of states, T is a finite set of events, $\Theta \subseteq (S \times T \times S)$ is the set of labeled state transitions, and s_0 is the initial state of TS.*

The pair (TS, F) where $F \subseteq S$ is the set of final states is a transition system with final states.

Definition 3 (Isomorphism of Transition Systems). *Let $TS = (S, T, \Theta, s_0)$ and $TS' = (S', T', \Theta', s_0')$ be two transition systems. TS and TS' are isomorphic if and only if $T = T'$ and there is a bijective function $\phi : S \to S'$ so that $\phi(s_0) = s_0'$ and $(s, t, s') \in \Theta$ if and only if $(\phi(s), t, \phi(s')) \in \Theta'$. If TS and TS' are isomorphic and ϕ is the related isomorphism, we write $TS \cong_\phi TS'$.*

Two transition systems with final states (TS, F) and (TS', F') are isomorphic if and only if there is a function ϕ so that $TS \cong_\phi TS'$ and $\phi(F) = F'$ holds.

Transition systems are able to showcase the behavior of elementary net systems.

Definition 4 (Reachability Graph). *Let $N = (P, T, A, m_0)$ be an elementary net system. We call $RS(N) = \{m \subseteq P | \sigma \in T^*, m_0 \xrightarrow{\sigma} m\}$ the set of reachable states. The reachability graph of N is the transition system $RG(N) = (RS(N), T, \Delta, m_0)$ defined by $(m, t, m') \in \Delta$ if and only if $m \in RS(N)$ and $m \xrightarrow{t} m' holds$.*

Let (N, C) be an elementary net system with final configurations, the reachability graph of (N, C) is the transition system with final states $RG(N, C) = (RG(N), C)$.

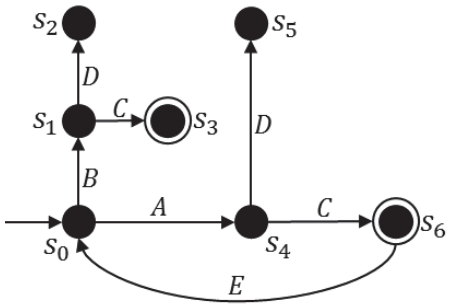


Fig. 3. A transition system with final states

Figure 3 depicts the reachability graph with final states of the net depicted in Figure 2. In figures a final state is enclosed by a circle. Starting from the initial state, both sequences B, C and A, C lead to a final state. As mentioned above, one of them is a deadlock, the other one enables the state transition E.

Definition 5 (Synthesis). *Let TS be a transition system. The synthesis problem is to construct an elementary net system N whose reachability graph is isomorphic to TS, i.e. there is a function ϕ so that $TS \cong_\phi RG(N)$ holds.*

Let (TS, F) be a transition system with final states. The synthesis problem is to construct an elementary net system with final configurations (N, C) whose reachability graph is isomorphic to (TS, F), i.e. there is a function ϕ so that $(TS, F) \cong_\phi RG(N, C)$.

In the following theorem, we prove that the synthesis of elementary net systems with final configurations consists of two parts: in a first step, we need to solve the synthesis problem for the underlying elementary net system without considering the final states. In a second step, we use the isomorphism between the transition system and the reachbility graph of the generated net to define the set of final configurations.

Theorem 1. *Let (TS, F) be a transition system with final states. There is a solution of the synthesis problem for (TS, F) if and only if there is a solution of the synthesis problem for TS.*

Proof. Let (TS, F) be a transition system with final states.

If there is a solution N of the synthesis problem for the transition system TS there is a function ϕ so that $RG(N) \cong_\phi TS$ holds. We define the set of final configurations as $\phi(F)$. Thus, $RG(N, \phi(F)) = (RG(N), \phi(F)) \cong_\phi (TS, F)$ holds. $(N, \phi(F))$ is a solution of the synthesis problem for (TS, F).

If there is a solution (N, C) for the synthesis problem for (TS, F) there is a function ϕ so that $RG(N, C) \cong_\phi (TS, F)$ holds. Of course, $RG(N) \cong_\phi (TS)$ holds. N is a solution for the synthesis problem for TS.

Altogether, there is a solution for (TS, F) if and only if there is a solution for TS.

3 Neat Places and Final Configurations

As stated in the introduction, we use elementary net systems with final configurations to model distributed systems. Of course, the more complex the system, the more complex the net system. Thus, final configurations of a net may be complex as well. To check if a net is in a final configuration, we need to compare the current (maybe distributed) configuration to the set of all final configurations. When modeling systems that are critical to safety, it may be necessary to immediately recognize if a system is in a final configuration. One particular simple possibility is to check a single place if a token on this place indicates, that the configuration is final. In this case, we call the place neat. Identifying neat places is of great value, especially if the final states of the underlying system are distributed, their number is huge, or they share different local variables confusingly. Furthermore, parts of a system may be not accessible at any time, i.e. the current state cannot be determined. If at least one local variable, modelled by one neat place, is accessible, other parts of the configuration are not relevant.

Definition 6 (Neat Place). *Let $(N, C) = (P, T, A, m_0, C)$ be an elementary net system with final configurations. A place $p \in P$ is neat, iff*

$$\forall m \in RS(N) : p \in m \Leftrightarrow m \in C.$$

As an example, we consider the elementary net system depicted in Figure 2. In this net neither place p_1 nor place p_3 is neat. The place p_1 is not neat, because there is the final configuration $\{p_2, p_3\}$ without p_1. The place p_3 is not neat, because firing the sequence of transitions A, D from the initial configuration leads to the configuration $\{p_1, p_3\}$. This configuration contains p_3, but this is not a final configuration. However, place p_2 is neat. Both sequences of transitions A, C and B, C mark place p_2 and lead to final configurations. The other two sequences A, D and B, D do not mark p_2 and do not lead to final configurations. After the occurrence of A, C, the same holds as soon as transition E occurs. Altogether, p_2 is only marked if the elementary net system is in a final configuration.

Fix a single place of an elementary net system. To check if this place is neat, we check the condition of Definition 6 for every state of the reachability graph. Fix an elementary net system. To decide if there is a neat place in this system, we do not need to check all of its places. In the remainder of this section, we introduce how to reveal a neat place using the theory of regions. In a first step, we decide if it is even possible, that there is a neat place for this specific elementary net system. In a second step, we check if this potential neat place is already a place of this net. If there exists a neat place, but it is not in the net, we show how to add this place.

We briefly recapitulate the theory of regions for the synthesis of elementary net systems, but refer the reader to [4] for a nice and complete introduction to region theory.

Definition 7 (Region). *Let (S, T, Δ, s_0) be a transition system. A region r is a set of states $r \subseteq S$ so that for all $t \in T$ one of the following situations hold:*

$$t \text{ enters } r, \text{ i.e. } \forall (s, t, s') \in \Delta \Rightarrow (s \notin r \wedge s' \in r)$$
$$t \text{ exits } r, \text{ i.e. } \forall (s, t, s') \in \Delta \Rightarrow (s \in r \wedge s' \notin r)$$
$$t \text{ does not cross the border of } r, \text{ i.e. } \forall (s, t, s') \in \Delta \Rightarrow (s \in r \Leftrightarrow s' \in r)$$

The main theorem of region theory states that every place of an elementary net system defines a region of its reachability graph. This region is the set of configurations including this place. This set is called the extension of the place.

Theorem 2. [5, 4] *Let N be a elementary net system $N = (P, T, A, m_0)$. For all $p \in P$ the extension of $[\![p]\!] = \{s \in RS(N) | p \in s\}$ is a region.*

For the proof we refer the reader to [5] or [4]. We consider the extension of place p_1 of Figure 2 in the transition system depicted in Figure 3 as an example. The extension of p_1 is the set of states $\{s_4, s_5, s_6\}$. This set is a region, because A enters, E exits, and all other events do not cross the border of this set. Like p_1, every other place of Figure 2 defines a region of the transition system depicted in Figure 3.

Our goal is to decide if there is a neat place for our specific elementary net system. On the one hand, according to Theorem 2, the extension of this place is a region. On

the other hand, if we fix a region, this region directly relates to a place, arcs and the initial marking which we could add to our elementary net system without changing its reachability graph. To construct a place p from a region r of a reachability graph $RG(N)$ of some net $N = (P, T, A, m_0)$, we add p to P, for every $t \in T$ we add (p, t) to A if t exits r and we add (t, p) to A if t enters r. We add p to m_0 only if $m_0 \in r$ holds. In this situation we say that p is defined by r. Summing up, the neat place we are looking for is defined by some region of the reachability graph of our elementary net system. Furthermore, this place is only marked at final configurations. The following theorem states that there is a neat place for an elementary net system if and only if the set of final states is a region.

Theorem 3. *Let (N, C) be an elementary net system with final configurations. There is a net (N', C) with a neat place so that $RG(N, C) \cong_\phi RG(N', C)$ holds if and only if C is a region of $RG(N, C)$.*

Proof. Let (N, C) be an elementary net system. If C is a region of $RG(N, C)$, construct a place p defined by a region r and add p to N to get (N', C). By definition of a region, $RG(N, C) \cong_\phi RG(N', C)$ holds. Furthermore, $[\![p]\!] = C$ holds and p is neat.

Assume C is not a region of $RG(N, C)$, but there is an elementary net system (N', C), $RG(N, C) \cong_\phi RG(N', C)$ with a neat place p. The place p is neat and by definition, $C = [\![p]\!]$ holds. According to Theorem 2, $[\![p]\!]$ is a region. This is a contradiction.

According to Theorem 3 it is easy to reveal a neat place. We consider the reachability graph of an elementary net system and check if the set of final states is a region. For example, we consider the transition system of Figure 3. The set of final states is a region, because transition C enters, E exits and all other transitions do not cross the border of this set. The place defined by this region has two adjacent arcs, i.e. one arc starting from transition C and one arc ending in transition E. This place is not marked, since the initial state of the transition system is not a final state. Altogether, p_2 of Figure 2 is neat. If the set of final states is a region, but defines a place which is not in the set of places of the elementary net system, we can simply add this place to the net. Since the new neat place is defined by a region, it will not change the reachability graph.

4 Neat Terms identifying Final Configurations

Whenever the set of final configurations is a region, it is easy, as discussed in Section 3, to generate a neat place. In this section, we introduce a heuristic to identify final configurations, although they do not form a region. We do this by considering terms over a subset of places which determine if an elementary net system is in a final configuration. We illustrate our idea by considering the transition system depicted in Figure 4, but this time we assume a different set of final states.

In Figure 4, the set of final states $\{s_5\}$ is not a region, because one event D enters while the other one does not. The related elementary net system is the net depicted in Figure 2. The set, $\{p_1, p_3\}$ is the final configuration for this example. This time, there is no neat place, so we need to check all five places to decide if the net is in its

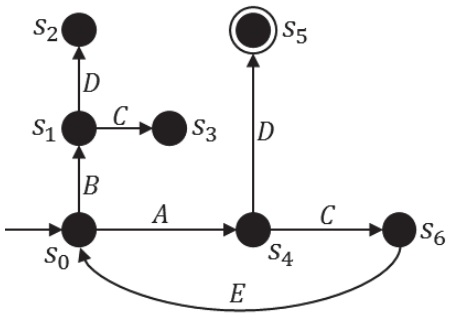
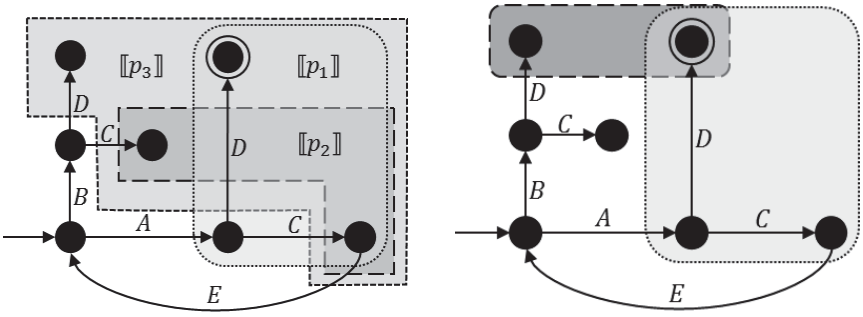


Fig. 4. A transition system with one final state.

final configuration. To avoid such a brute-force testing we will identify a term of places representing this final configuration. According to the definition of a neat place, we call such a term a neat term in the following.

To illustrate the idea of neat terms, we consider Figure 5a, the transition system of Figure 4, and depict the extensions of places p_1, p_2, and p_3 of Figure 2 by dashed frames. The intersection of $[\![p_1]\!]$ and $[\![p_3]\!]$ is $\{s_5, s_6\}$. The set difference of $\{s_5, s_6\}$ and $[\![p_2]\!]$ is $\{s_5\}$, i.e. the final state in this example. In other words, $\{p_5\}$ is not a region, but $\{p_5\} = ([\![p_1]\!] \cap [\![p_3]\!]) \cap [\![p_2]\!]^C$ holds. We conclude, the net is in a final configuration if p_1 and p_3 are marked and additionally p_2 is not. In this example, $p_1 \wedge \neg p_2 \wedge p_3$ is a neat term. Thus, to check if the net is in a final configuration, we no longer have to check all five places, but only three.



(a) Extensions of p_1, p_2, and p_3.

(b) Extension of p_1 and a region $\{s_2, s_5\}$.

Fig. 5. Extensions in the transition system.

We are able to construct an even smaller neat term if we consider the region $\{s_2, s_5\}$ depicted in Figure 5b. This set is a region, because only event D enters, while all others do not cross the border of this set. Utilizing this region we get $\{s_2, s_5\} \cap [\![p_1]\!] = \{s_5\}$, the final marking of our example. We extend the net of Figure 2 by a place related to

the region $\{s_2, s_5\}$. The resulting elementary net system is depicted in Figure 6. In this net, by adding one additional place, we get the neat term $p_1 \wedge p_5$ using only two places. If and only if these two places are marked, the net is in its original final configuration, i.e. p_1 and p_3 are marked, as p_0, p_2, p_4 are not. Altogether, the places p_0, p_2, p_3 and p_4 are negligible to decide if the net is in its final configuration.

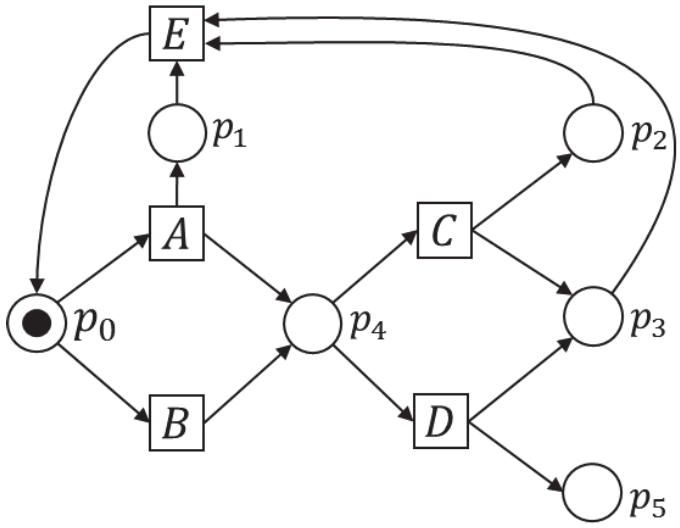


Fig. 6. An elementary net system with an additional place for the region $\{s_2, s_5\}$ of Figure 5b, the final configuration $\{p_1, p_3, p_5\}$, and the neat term $p_1 \wedge p_5$.

Every place may be some boolean variable of a neat term. Furthermore, neat terms consist of $\wedge$, $\vee$, and $\neg$ operators. Any term is neat, if and only if the related extensions, translating the operators to set operators $\cap$, $\cup$, and ^{C}, form the set of final configurations. As stated above, if we have a distributed system and we are not able to determine (at run-time) the configuration of this system, neat terms may aid the problem to decide if the system is in a final configuration or not. We consider the extensions of places that may be accessed and try to add additional places defined by regions to represent the set of final states by unions, intersections, and complements.

5 Conclusion and Future Work

We presented the definition of transition systems with final states and the definition of elementary net systems with final configurations. We proved, that the related synthesis problem is solvable if and only if the same problem ignoring the final states is solvable. In Section 3, we presented a definition of so called neat places. A place is neat if it is marked exactly at final configurations. With the help of neat places a complex set of final states can be represented efficiently. If, for a chosen elementary net system, there

is no neat place, we presented a heuristic to identify terms of places to indicate final configurations. For neat places as well as for neat terms we explained how to extend the set of places of an elementary net system if this leads to a net with a more simple set of final configurations, in a sense that a neat place or a smaller neat term can be found.

In future work we will implement a deduced synthesis algorithm constructing neat terms from the set of minimal regions of a transition system with final states. In a second step, a user will be allowed to mark a set of places as not accessible. Thus, this set is excluded from the construction of a possible neat term.

Of course, it is possible to extend the class of considered Petri nets. We can apply the same approach to Petri nets with weighted arcs, as long as their reachability graph is finite. An open research question is to state if the same idea carries over to general place/transition nets using abstract representations of the related infinite reachability graphs.

References

[1] van der Aalst, W. M. P.: The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers* 8(1), World Scientific, 1998, 21-66

[2] van der Aalst, W. M. P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, 2011

[3] van der Aalst, W. M. P.; Adriansyah A.; Dongen, B.; Katoen, J. P.; König, B.: Causal Nets: A modeling Language Tailored towards Process Discovery. *Proceedings of CONCUR 2011*, Springer, 2011, 28-42

[4] Badouel, E.; Bernardinello, L.; Darondeau, P.: Petri Net Synthesis. *Texts in Theoretical Computer Science* Springer, 2015

[5] Ehrenfeucht, A.; Rozenberg, G.: Partial (set) 2-structures; Part 1 *Acta Informatica* 27(4), 1990, 315-342

[6] Ehrenfeucht, A.; Rozenberg, G.: Partial (set) 2-structures; Part 2 *Acta Informatica* 27(4), 1990, 343-368

[7] Hopcroft, J. E.; Motwani, R.; Ullman, J. D.: Introduction to Automata Theory, Languages, and Computation. *Texts in Theoretical Computer Sience* Addison-Wesley, 2006

[8] ter Hofstede, A.H.M; van der Aalst, W. M. P.; Adams M; Russel N.: Modern Business Process Automation: YAWL and Its Support Environment. Springer, 2010

[9] Nielsen, M.; Rozenberg, G.; Thiagarajan, P.S.: Elementary transition systems. *Theoretical Computer Science* 96(1), Elsevier, 1992, 3-33

[10] OMG. Business Process Model and Notation (BPMN). Object Management Group, 2016

[11] Scheer, A.W.: Business Process Engineering, Reference Models for Industrial Enterprises. Springer, 1994

[12] UML. Unified Modeling Language (UML). Object Management Group, 2016