Compact Regions for Place/Transition Nets

Robin Bergenthum

Department of Software Engineering and Theory of Programming,
FernUniversität in Hagen
robin.bergenthum@fernuni-hagen.de

Abstract. This paper presents compact regions to synthesize a Petri net from a partial language. We synthesize a Petri net using the theory of regions. Let there be a partial language, every region definition provides an inequality system and a solution of this system is called a region. Every region defines a valid place where a place is valid if it enables every word of the partial language. The new compact region definition relies on compact tokenflows. Compact tokenflows are a very efficient behavioral model for the partial language of Petri nets [3, 4]. Compact regions will lead to faster synthesis algorithms computing smaller Petri nets solving the synthesis problem.

1 Introduction

We synthesize a place/transition net (p/t-net) from a partial language, i.e. a set of labeled partial orders (lpos) using the theory of regions [1, 2, 5]. Every type of region is based on a behavioral model. Of course, different behavioral models result in different region definitions, but the concepts in language based region theory are always the same [6, 7].

A place of a Petri net is valid for a specified language if it enables all words of the language. If we execute the language in the net, the firing rule is always satisfied for every valid place. Fix some type of behavioral model, fix a place, and fix a language: there is an inequality system so that there is a solution of this system if and only if the place is valid. The main idea of region theory is to consider the place as a variable in this inequality system. All at once, we are able to solve the synthesis problem, because we are able to calculate the set of all valid places.

The main steps of a region based synthesis algorithm are: Let there be a language over a set of labels and build a transition for every label. Choose a behavioral model and build the corresponding inequality system to check if an arbitrary place is valid. Consider the place to be a variable and calculate the basis of the solution space. For every solution in the basis add a place (and its arcs) to the set of transitions. The resulting net solves the synthesis problem, because of two arguments: First, every place of the constructed net is valid so that the Petri net enables all words of the language. Second, every additional place (not in the net) is either a linear combination of added places or is not valid. Altogether, the constructed net acts as specified or there is no such net.

In this synthesis algorithm, different behavioral models result in different inequality systems describing valid places. The run-time, as well as the size of the calculated Petri net, are mainly determined by the size of this inequality system. In the literature, there are two different types of regions considering partial order behavior of Petri nets. Both definitions have plenty disadvantages in most examples and applications. A transition region [7] belongs to the behavioral model of enabled cuts [8]. A place is valid if every set of unordered events of a specified language is enabled to occur after the occurrence of its prefix. The enabled cuts inequality system states that every prefix (i.e. its Parikh vector plus initial marking) produces enough tokens to enable the next maximal step. Therefore, transition regions yield an inequality for every cut of a language. A tokenflow region [6] belongs to the behavioral model of tokenflows [9]. A place is valid if there is a valid distribution of tokens along the transitive order relation of the specified language. Such a distribution of tokens needs to respect the firing rule of Petri nets. The tokenflow inequality system demands that every event does not produce too many tokens and every event receives enough tokens to occur. Therefore, there is a variable for every arc of the specified language as well as two inequalities for every event.

In practical applications, partial languages have many cuts and many arcs. The enabled cuts inequality system as well as the tokenflow inequality system is huge. Therefore, both existing synthesis algorithms have impracticable run-time. Paper [7] states, the more concurrency is in the language the worse is a transition region algorithm. The more dependency is in the language the worse is a tokenflow region algorithm.

In this short paper we present a new definition of compact region. This definition is related to the most recent behavioral model for partial languages presented in [3, 4]. We will show that compact regions lead to much smaller region inequality systems. The size of these systems is related to the size of the Hasse-diagrams of the specified language. Furthermore, the reduced size leads to smaller, i.e. nicer, Petri nets.

2 Preliminaries

Let f be a function and B be a subset of its domain A. We write $f|_B$ to denote the restriction of f to B. A function $m : A \to \mathbb{N}$ is called a multiset. We write $m = \sum_{a \in A} m(a) \cdot a$ to denote multiplicities of elements in m. Let $m' : A \to \mathbb{N}$ be another multiset. We write $m \geq m'$ if $\forall a \in A : m(a) \geq m'(a)$ holds. We denote the transitive closure of an acyclic and finite relation $\prec$ by $\prec^*$. We denote the skeleton of $\prec$ by $\prec^\diamond$. The skeleton of $\prec$ is the smallest relation $\lhd$ so that $\lhd^* = \prec^*$ holds. We model concurrent or distributed systems by place/transition nets [10].

Definition 1. *A place/transition net (p/t-net) is a tuple (P, T, W) where P is a finite set of places, T is a finite set of transitions so that $P \cap T = \emptyset$ holds, and $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a multiset of arcs. A marking of (P, T, W) is a multiset $m : P \to \mathbb{N}$. Let m_0 be a marking. We call $N = (P, T, W, m_0)$ a marked p/t-net and m_0 the initial marking of N.*

Let t be a transition. We denote $\circ t = \sum_{p \in P} W(p, t) \cdot p$ the weighted preset of t. We denote $t\circ = \sum_{p \in P} W(t, p) \cdot p$ the weighted postset of t. A transition t can fire at marking m if $m \geq \circ t$ holds. If t fires, m changes to $m' = m - \circ t + t\circ$. The most famous behavioral model for partial order behavior of Petri nets is a process [11]. A process is a Petri net modeling one single run of a marked p/t-net.

Definition 2. *A process net is a tuple $O = (C, E, F)$ where C is a finite set of conditions, E is a finite set of events so that $C \cap E = \emptyset$ holds, and $F \subset (C \times E) \cup (E \times C)$*

is a set of arcs. The graph $(C \cup E, F)$ is acyclic and does not branch at conditions, i.e. every condition has at most one predecessor and at most one successor.

Let $N = (P, T, W, m_0)$ be a marked p/t-net. A process is a pair (O, ρ) where $O = (C, E, F)$ is a process net and $\rho : (C \cup E) \to (P \cup T)$ is a labeling function. (O, ρ) is a process of N iff the following conditions hold:

(1) $\rho(C) \subseteq P$, $\rho(E) \subseteq T$,
(2) $\forall e \in E : \circ\rho(e) = \sum_{p \in P} |\{(c, e) \in F | \rho(c) = p\}| \cdot p$,
(3) $\forall e \in E : \rho(e)\circ = \sum_{p \in P} |\{(e, c) \in F | \rho(c) = p\}| \cdot p$,
(4) $\forall p \in P : |\{c \in C | \forall e \in E : (e, c) \notin F, \rho(c) = p\}| = m_0(p)$.

Every process of a Petri net N relates to a labeled partial order. The set of labeled partial orders induced by all processes of N is the partial language of N.

Definition 3. *A labeled partial order (lpo) is a triple $lpo = (V, <, l)$ where V is a finite set of events, $< \subseteq V \times V$ is a transitive and irreflexive relation, and the labeling function $l : V \to T$ assigns a label to every event.*

Definition 4. *Let $K = (C, E, F, \rho)$ be a process of a marked p/t-net N. The lpo $(E, F^*|_{E \times E}, \rho|_E)$ is the process lpo of K. Let N be a marked p/t-net and $L^{\Pi}(N)$ be the set of all process lpos of N. $L(N) = \{(E, <, l) | (E, <, l)$ an lpo, $(E, \prec, l) \in L^{\Pi}(N), \prec \subseteq <\}$ is the partial language of N.*

We specify behavior of a system by example runs. An example run is a set of events with an acyclic relation. Of course, every example run relates to an lpo and we can extend the partial language of Petri nets to example runs.

Definition 5. *A triple $run = (V, \prec, l)$ is an example run if $(V, \prec^*, l)$ is a labeled partial order. A finite set of example runs is a specification. Let $run = (V, \prec, l)$ be an example run. We define $run^* = (V, \prec^*, l)$ the lpo and $run^\diamond = (V, \prec^\diamond, l)$ the compact lpo (cpo) of run.*

Definition 6. *Let N be a marked p/t-net and $S = \{run_1, \ldots, run_n\}$ be a specification. We write $S \subseteq L(N)$ iff $\{run_1^*, \ldots, run_n^*\} \subseteq L(N)$ holds.*

Synthesis is to construct a Petri net so that its behavior matches the specification. If there is no such net, we construct a net so that its behavior includes the specification and has minimal additional behavior.

Definition 7. *Let S be a specification, the synthesis problem is to compute a marked p/t-net N so that the following conditions hold: $S \subseteq L(N)$ and for all marked p/t-nets $N' : L(N) \backslash L(N') \neq \emptyset \Longrightarrow S \not\subseteq L(N')$.*

3 Compact Regions

We synthesize a p/t-net from a partial language applying the theory of regions. We construct a transition for every label of the specification and get a p/t-net without places. The language of this net includes arbitrary behavior. Obviously, we need to add places and arcs to restrict the behavior of such an initial net. Of course, we only add places that do not prohibit the specification.

Definition 8. *Let S be a specification and $N = (P, T, W, m_0)$ be a marked p/t-net. A place $p \in P$ is called feasible for S iff $S \subseteq L((\{p\}, T, W|_{(\{p\}\times T)\cup(T\times\{p\})}, m_0(p)))$.*

Let S be a specification and $N = (\{p\}, T, W, m_0)$ be a marked one-place p/t-net. We call N a feasible p/t-net for S if p is feasible for S.

Corollary 1. *Let S be a specification and T be its set of labels. Let $\{(\{p_1\}, T, W_1, m_1), \ldots, (\{p_n\}, T, W_n, m_n)\}$ be a set of feasible p/t-nets and $N = (\bigcup_i p_i, T, \bigcup_i W_i, \bigcup_i m_i)$ be their union. Of course, every place of N is feasible and $S \subseteq L(N)$ holds.*

In region theory it is well known that the following theorem holds [5].

Theorem 1. *Let S be a specification and T be its set of labels. The infinite p/t-net which is the union of all feasible p/t-nets is a solution of the synthesis problem.*

To discover feasible places, we use compact tokenflows as behavioral model [3, 4].

Definition 9. *Let $N = (P, T, W, m_0)$ be a p/t-net and $run = (V, \prec, l)$ be an example run so that $l(V) \subseteq T$ holds. We denote $\prec^\diamond = \lhd$. A compact tokenflow is a function $x : (V \cup \lhd) \to \mathbb{N}$. x is compact valid for $p \in P$ iff the following conditions hold:*

(i) $\forall\, v \in V$: $x(v) + \sum_{v' \lhd v} x(v', v) \geq W(p, l(v))$,

(ii) $\forall\, v \in V$: $\sum_{v \lhd v'} x(v, v') \leq x(v) + \sum_{v' \lhd v} x(v', v) - W(p, l(v)) + W(l(v), p)$,

(iii) $\sum_{v \in V} x(v) \leq m_0(p)$.

run is compact valid for N iff there is a compact valid tokenflow for every $p \in P$.

Papers [3, 4] prove that the partial language of a marked p/t-net is the set of its compact valid example runs.

Theorem 2. *The language of a marked p/t-net is its set of compact valid example runs.*

At this point we are able to state the main contribution of this short paper. We take advantage of compact tokenflows and define the notion of a compact region.

Definition 10. *Let $S = \{(V_1, \prec_1, l_1), \ldots, (V_n, \prec_n, l_n)\}$ be a specification, T be its set of labels, and p be a place. We denote $\prec_i^\diamond = \lhd_i$.*

A function $r : (\bigcup_i (V_i \cup \lhd_i) \cup (T \times \{p\}) \cup (\{p\} \times T) \cup \{p\}) \to \mathbb{N}$ is a compact region for S iff $\forall i \in \mathbb{N} : r|_{\{V_i \cup \lhd_i\}}$ is compact valid for p in $(\{p\}, T, r|_{(T\times\{p\})\cup(\{p\}\times T)}, r(p))$.

Theorem 3. *Let S be a specification and T be its set of labels. Every compact region r for S defines a feasible p/t-net $N_r = (\{p\}, T, W, m_0)$ and vice versa.*

Proof. Let r be a compact region. For every example run in S there is a valid compact tokenflow $r|_{\{V_i \cup \lhd_i\}}$ of p in $N_r = (\{p\}, T, r|_{(T\times\{p\})\cup(\{p\}\times T)}, r(p))$. Of course, $S \subseteq L(N_r)$ holds and N_r is feasible.
Let $N = (\{p\}, T, W, m_0)$ be a feasible p/t-net so that $S \subseteq L(N_r)$ holds. There is a valid compact tokenflow r_i for every example run of S. Obviously, the union $r = \bigcup_i r_i \cup W \cup m_0$ is a compact region.

Altogether, we are able to express the set of all feasible p/t-nets by a single inequality system. In this system there is a variable for every element in the domain of a compact region, i.e. one variable for every event, another variable for every skeleton arc, two variables for every label, and a single variable for the initial marking. The complete inequality system is built from the inequalities defined in Definition 9. According to (i) and (ii) there are two inequalities for every event of the specification. According to (iii) there is another inequality for every example run. The set of positive integer solutions of this inequality system is the set of all feasible nets. We call this inequality system the compact region inequality system. The union of positive integer basis solutions of the compact region inequality system is a solution of the synthesis problem.

The transition region inequality system [7] and the tokenflow region inequality system [6] are defined just like a compact region inequality system. The tokenflow region inequality system has two additional variables for every transitive arc of the specification. For most examples the size of the tokenflow system is quadratic in the size of the compact system. The transition region inequality system has one inequality for every cut of the specification. Note, the number of cuts may be exponential in the number of events and the number of cuts is always bigger than the number of skeleton arcs. Altogether, the compact system is always smaller than the other two inequality systems. Compact regions are the most efficient definition of a region of a partial language. Right now, I am implementing compact region synthesis in a tool called MoPeBs Alpaca. First results are very promising and match the theoretical considerations. Synthesis is much faster and the compact solution space leads to nets having fewer places. Of course, this is provisional. Comprehensive implementation and run-time tests is future work.

References

[1] Ehrenfeucht, A.;Rozenberg, G.: *Partial (Set) 2-Structures. Part I: Basic Notions and the Representation Problem*. Acta Inf. 27 (4), 315–342, 1990.

[2] Ehrenfeucht, A.;Rozenberg, G.: *Partial (Set) 2-Structures. Part II: State Spaces of Concurrent Systems*. Acta Inf. 27 (4), 343–368, 1990.

[3] Bergenthum, R.: *Verifikation von halbgeordneten Abläufen in Petrinetzen*. PhD in computer science, Library of the University of Hagen, 2013.

[4] Bergenthum, R.; Lorenz, R.: *Verification of Scenarios in Petri Nets Using Compact Tokenflows*. Fundamenta Informaticae 137, 117–142, IOS Press, 2015.

[5] Badouel, E.; Darondeau P.: *Theory of Regions*. Lectures on Petri Nets, LNCS 1491, 529–586, Springer, 1998.

[6] Bergenthum, R.; Desel, J.; Lorenz, R.; Mauser, S.: *Synthesis of Petri Nets from Finite Partial Languages*. Fundamenta Informaticae 88, 437–468, IOS Press, 2008.

[7] Bergenthum, R.; Desel, J.; Mauser, S.: *Comparison of Different Algorithms to Synthesize a Petri Net from a Partial Language*. ToPNoC 3, LNCS 5800, 216–243, Springer, 2009.

[8] Grabowski, J.: *On partial languages*. Fundamenta Informaticae 4, 427–498, IOS Press, 1981.

[9] Bergenthum, R.; Desel, J.; Juhs, G.; Lorenz, R.; Mauser, S.: *Executability of Scenarios in Petri Nets.*. Theoretical Computer Science 410 (12-13), 1190–1216, Elsevier, 2009.

[10] Desel, J.; Reisig, W.: *Place/Transition Petri Nets*. Lectures on Petri Nets, LNCS 1491, 122–173, Springer, 1998.

[11] Goltz, U.; Reisig, W.: *Processes of Place/Transition-Nets*. Automata Languages and Programming 154, 264–277, Springer, 1983.