Andrei KARATKEVICH
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Analysis of Parallel Discrete Systems: Persistent Sets and Concurrent Simulation

Ph.D. Andrei KARATKEVICH

Received the Ph.D. degree from Belarusian State University of Informatics and Radioelectronics in 1998. Since 2000 he is an assistant professor at Technical University of Zielona Góra (since 2001 - University of Zielona Góra). His current research interests include Petri net theory and its applications, especially to problems related to analysis and verification of parallel or distributed control systems.

e-mail: A.Karatkevich@iie.uz.zgora.pl

Abstract

This paper is focused on problem of analysis of parallel discrete systems, which can be described by Petri nets. The general analysis approach considered in the paper is optimal simulation, allowing checking properties of the system by constructing reduced state spaces. Two well-known methods of such analysis are based on persistent sets and concurrent simulation, correspondingly. Here we discuss possibility of combination between those methods and describe an algorithm using both ideas. Combination with decomposition approach is also discussed.

Keywords: parallel discrete systems, Petri nets, simulation, state exploration.

Analiza współbieżnych systemów dyskretnych: uparte zbiory i symulacja współbieżna

Streszczenie

Tematem pracy jest problem analizy współbieżnych systemów dyskretnych, które mogą być opisane sieciami Petriego. Ogólnym podejściem do analizy, omawianym w artykule, jest symulacja optymalna, która pozwala sprawdzać właściwości sieci poprzez konstruowanie zredukowanej przestrzeni osiągalności. Dwie znane metody takiej analizy bazują na tak zwanych „upartych zbiorach" oraz na współbieżnej symulacji. W pracy przedstawiono wykorzystanie tych idei w ramach jednej metody. Omówione są też możliwości i zalety połączenia takiej metody z wykorzystaniem dekompozycji.

Słowa kluczowe: współbieżne systemy dyskretne, sieci Petriego, symulacja, eksploracja stanów.

1. Introduction

The basic model of digital device, used in theory of automata and practice of digital design, is Finite State Machine. However, very often design of discrete systems, in particular control systems, requires generalization of automata description allowing parallelism. There are two main ways of parallel automata description; one is based on FSM networks, another – on Petri nets.

Petri nets [8] are a formal model describing parallel asynchronous discrete systems. Petri nets are used in modeling and design of digital devices, especially logic controllers. There exist several languages for specification of control algorithms, based on Petri nets formalism.

Analysis and verification of parallel systems such as Petri nets is much more complex task, than analogous tasks for pure sequential descriptions. It is caused by the fact that number of reachable global states of a parallel system depends exponentially on its size (the state explosion problem), so direct managing of such system's state space often turns to be practically impossible. On the other hand, for the parallel systems there exist such analysis tasks which do not exist for the sequential systems.

There are multiple methods and algorithms of analysis of parallel systems. Some of them avoid state exploration at all; some others reduce it, searching only a part of state space [3]. There are two main approaches to reduced state space construction. One is based on selecting at each step of simulation a subset of possible (active) transitions to be simulated; it is generally known as *persistent set method* [10], its most elaborate technique is *stubborn set method* [9]. Another one is based on the idea of concurrent simulation of active transitions [4, 5]. Other known approaches are listed in [3].

Each of the mentioned approaches has remarkable advantages. Is it possible, however, to combine those advantages within one method? Direct combination of those two approaches seems to be impossible, because the main ideas are almost opposite: generally, if two non-conflicting transitions are simultaneously enabled, the first way supposes that their firing will be simulated one-by-one, and the second – simultaneously.

But the maximal concurrent simulation approach is not sufficiently expressive [4, 5]; for example, it is easy to show, that, being directly applied, it may fail to detect some deadlocks. So, maybe the concurrent simulation approach can be improved by the idea of persistent sets? We claim that the answer is positive. In [6] a particular case of this possibility is presented; here we give a generalization of the idea described there.

2. Petri nets

A Petri net [8] can be presented as a bipartite oriented graph with two kinds of nodes: *places* and *transitions*. Places and transitions are connected by arcs. A *marking* is an allocation of *tokens* which can be assigned to places. Allocation and number of tokens in a net can change by transition firing, which is regulated by simple rules [8]. In modeling and design of discrete control systems the interpreted Petri nets [1] are used, being a Petri-net-based models enhanced by the elements making possible communication of the model with the outer world (conditions and events). The important properties of Petri nets are *liveness* and *safeness* [6]. A *deadlock* is a reachable marking in which no transition can fire. Deadlock detection is one of the theoretically and practically important tasks of analysis of Petri nets. Detailed definitions and notations are not presented here because of lack of space; see [4, 8, 10, 11].

In fig. 1 a Petri net is presented, being a part of specification of a control algorithm [2].

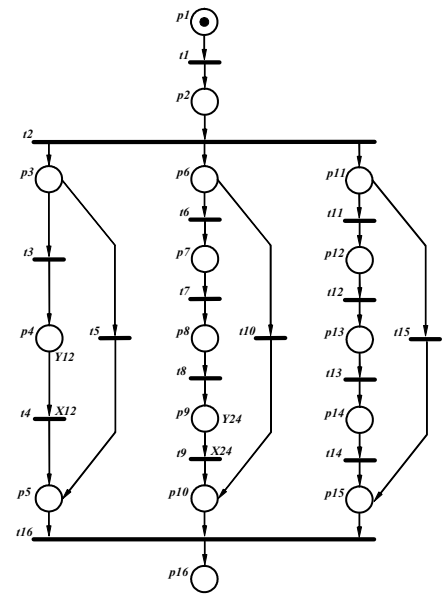

Fig. 1. A Petri net
Rys. 1. Sieć Petriego

3. Concurrent simulation and persistent sets

The next affirmations present a possibility of uniting of the mentioned approaches.

Lemma 1. Let M be a global state, $Mt_1M_1t_2M_2 \dots t_nM_n$ be a sequence in the reachability graph, t is a transition enabled in all the states $MM_1 \dots M_n$, and there is no such transition t_i in the sequence that ${}^\bullet t_i \cap t_i^\bullet \cap {}^\bullet t \neq \varnothing$. Then the sequence $tt_1t_2 \dots t_n$ is enabled at M and leads to the same marking M' as the sequence $t_1t_2 \dots t_nt$.

Proof. The effect of the firing of a transition at a marking is an addition of an integer valued vector to this marking. Vector addition is commutative, so if two sequences of transition firing are enabled at the same state and differ only in order of the transitions, the resulting marking is the same. So, it is enough to prove that the sequence $tt_1t_2 \dots t_n$ is enabled at M.

The proof proceeds by induction on n. Let $n=1$. Suppose that t disables t_1. Then t and t_1 share an input place p, which has 1 token at M. But as far as t_1 does not disable t, $p \in t_1^\bullet$, which contradicts the assumption ${}^\bullet t_i \cap t_i^\bullet \cap {}^\bullet t \neq \varnothing$. So, t and t_1 do not disable each other, and for $n=1$ the lemma holds.

Now, assume that the lemma holds for every sequence of length $(n-1)>0$, and let us prove that it holds for a sequence of length n. According to the inductive hypothesis, sequence $tt_1t_2 \dots t_{n-1}$ is enabled at M. It leads to the same state M' as $t_1t_2 \dots t_{n-1}t$. Firing of t at M_{n-1} does not disable t_n, which follows from the same proof as at the first step of induction. Hence the sequence $Mt_1t_2 \dots t_{n-1}ttM''$ exists. From the above also the sequence $Mt_1t_2 \dots t_{n-1}ttM''t_nM'$ exists, so the lemma holds for n. This together with the inductive hypothesis proves the lemma.

Theorem 1. Let $PN = (P,T,F,M_0)$ be a Petri net, let $U = (T_{P1}, T_{P2}, \dots T_{Pn})$ be a set of non-intersecting persistent sets at M_0. Simulate at M_0 every step $A_k = \{t_1^k, t_2^k, \dots, t_n^k\}$ such that $t_i^k \in T_{Pi}$. Repeat the operation for every newly obtained marking. Every deadlock reachable from M_0 in PN will be reached by such search.

Proof. Let D be a reachable deadlock. From Theorem 4 in [10], there is a sequence $M_0\, t_1t_2 \dots t_m\, D$ in the reduced reachability graph (RRG) created with the persistent set approach (also if at every step $T_P \in U$). As far as no transition is enabled at D and no transition outside a persistent set can disable a transition in the persistent set, for every $T_{Pi} \in U$ there is a transition t_i belonging to the sequence, such that all the previous transitions in the sequence are independent in respect of it; all such transitions (belonging to different persistent sets) are mutually independent. From Lemma 1, every such transition can be moved to the beginning of the sequence, and the sequence will remain enabled and will lead to the same marking. Then there exists (in the full reachability graph, but also, as it is easy to show, in an RRG) a sequence $M_0t_1't_2' \dots t_n'M_1\sigma D$ being an interleaving of $M_0\, t_1t_2 \dots t_m\, D$, such that there is step $A_1=\{t_1', t_2', \dots t_n'\}$, which will be simulated at M_0. So, $M_0A_1M_1$. If $M_1 \neq D$, repeat for the sequence $M_1\sigma D$ the same reasoning, as was applied above to $M_0\, t_1t_2 \dots t_m\, D$. $|\sigma|<m$, hence D will be reached in a finite number of steps.

4. Description of the proposed method

The proposed approach can be implemented in the next algorithm, which constructs an RRG for given Petri net $PN=(P,T,F,M_0)$.

Algorithm 1

1. Introduce the initial marking M_0 as a node and tag it "new".
2. While "new" markings exist, do the following:
   a. Select a new marking M. $Q:=\varnothing$, $R:=\varnothing$, $i:=0$. While $enabled(M) \setminus (Q \cup R) \neq \varnothing$, do the next.
      i. $i:=i+1$. Select $t_i \in (enabled(M) \setminus (Q \cup R))$.
      ii. Calculate T_{Si} such that $t_i \in T_{Si}$. If $T_{Si} \cap Q \neq \varnothing$, then $R:=R \cup \{t_i\}$ and $i:=i-1$; else $Q:=Q \cup T_{Si}$.
   b. For $j=1 \dots i$: $T_{Pi} := T_{Si} \cap enabled(M)$.
   c. Let A be the set of all steps $A_k = \{t_1^k, t_2^k, \dots, t_n^k\}$ such that $\$\, t_i^k \in T_{Pi}$.
   d. For every pair of transitions t, t' such that $t \in A_k \in A$, and t and t' are independent at M: $A_k := A_k \cup \{t'\}$.
   e. For every $A_k \in A$:
      i. Obtain the marking M' such that MA_kM'.
      ii. Introduce M' as a node, introduce an arc with label A_k from M to M', and tag M' "new".
   f. Remove label "new" from M.

It follows from Theorem 1, that the reachability graph generated by Algorithm 1 contains all the reachable deadlocks of the net. Examples of applying Algorithm 1 to the Petri nets see in Fig. 2, 3.

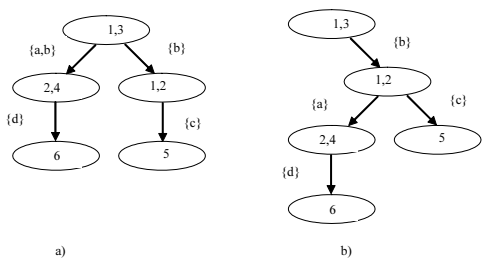

Fig. 2. Graphs of concurrent simulation of the biggest connected component of the net shown in Fig. 3a, according to OPT (a) and Algorithm 1 (b)
Rys. 2. Grafy współbieżnej symulacji największej spójnej komponenty sieci, pokazanej na rys. 3a - wg. OPT (a) i algorytmu 1 (b)

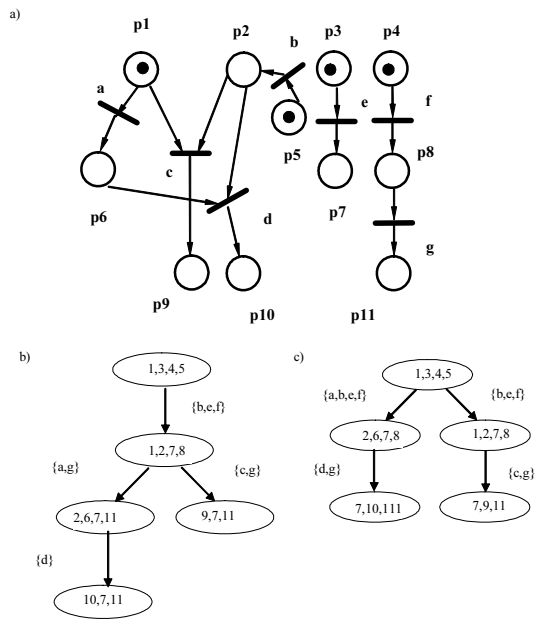

Fig. 3. A Petri net (Fig. 1.2 from [4]) (a), graph of concurrent simulation according to Algorithm (b) and graph of concurrent simulation OPT (c)
Rys. 3. Sieć Petriego (rys. 1.2 z [4]) (a), graf współbieżnej symulacji wg. algorytmu 1 (b) i graf współbieżnej symulacji OPT (c)

It is interesting to compare Algorithm 1 with the method of concurrent simulation by R. Janicki i M. Koutny, which is described by its authors as *the* optimal simulation (OPT) [4].

Examples of the graphs generated by OPT are shown in Fig. 2a, 3c. In some cases the graphs created with those methods are identical (as in Fig. 4).

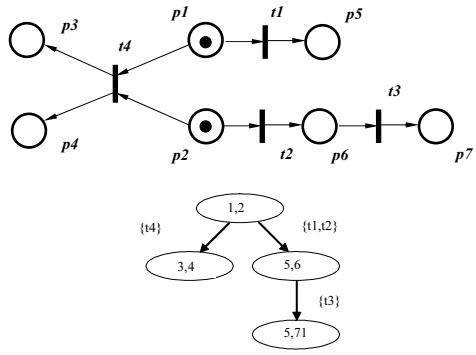

Fig. 4. A Petri net (Fig. 2.1 from [4]) and graph of concurrent simulation
Rys. 4. Sieć Petriego (rys. 2.1 z [4]) i graf współbieżnej symulacji

There are the next essential differences between Algorithm 1 and OPT.

- Generally OPT generates shorter firing sequences, but Algorithm 1 simulates less number of transition firings.
- OPT is defined only for the state machine decomposable (SMD) nets [4], Algorithm 1 works for any ordinary Petri net.
- In general case in reachability graph generated by OPT there may be different nodes for the same markings. Algorithm 1 follows the pattern of the algorithm of full reachability graph generation [8], and it constructs the graph in which not more than one node corresponds a marking.

5. Alternative branching, state explosion and decomposition

Concurrent simulation cannot avoid state explosion in case of nets with many alternative branchings, because such simulation would require checking of all the mutual combinations of alternative variants in all the parallel branches. Consider a net with r parallel branches, each consisting of n alternative branches of length q. Then complete reachability graph would consist of $(nq)^r$ nodes, RRG with maximal concurrent simulation – of $n^r q$ nodes, and RRG created by stubborn set method - of nqr nodes. That means that comparative effectiveness of those methods does not depend on the length of branches but does depend on the extent of alternative branching.

But this problem seems to be in certain respect similar to the problem of avoiding interleaving. Different ordering of independent transitions firing leads to the same result, so why consider all the mutual combinations? That is the background of persistent set approach. On the other hand, sometimes the final result (i.e. a deadlock) does not depend on the alternative choices in the parallel branches of a system. In the terms of blocks as defined in [11], it can be formulated as follows: the choices made inside the blocks affect the behavior of the rest of the system only through the terminal states of the blocks. Choices made inside the parallel blocks are important for the global behavior, only if they lead to different combinations of states of output poles. Why then we should consider all mutual combinations of those choices?

The net in Fig. 1 has 3 parallel branches, each of which consists of 2 alternative branches, beginning and ending up at the same place. Such *two-pole blocks* are typical for structures of control and computational algorithms [12]. The concurrent simulation approach, as it has been described so far, would check 2x3=6 variants of execution, but there is only one reachable deadlock, and intuitively it seems to be clear, that there is no sense to check all those variants.

The idea of blocks and block decomposition together with the idea of concurrent simulation allows developing analysis method excluding such checking. The first attempt to create such method is presented in [7] (this variant does not use the persistent set approach directly); deeper research in this direction is a prospective topic of future work.

6. Final notes

In this paper we concentrate on one (among many others), however very important in engineering applications, task of analysis of Petri nets – the task of deadlock detection. This task is essential practically, because for a system to be designed it is important to check whether and how it can be blocked and whether it can reach every specified terminal state. The task is also convenient for theoretical research, because it allows comparing easily and in evident way various analysis methods. But of course, other analysis tasks such as checking of liveness and safeness, are also of great importance. The described approach can be used (with some modifications) for liveness checking; its application to other analysis tasks requires additional research.

Summarizing, we can state, that the presented approach allows performing certain analysis of parallel systems by partial simulation using simpler sets of firing sequences, than the known methods. Additional advantage of the approach is possibility of its parallel implementation (different blocks of decomposed system can be analyzed simultaneously. Algorithm 1 does not present this possibility, being purely sequential; examples of parallel analysis algorithm see in [7]). So the approach seems to be useful for verification of big parallel systems.

The work is supported by Polish State Committee for Scientific Research (KBN) grant № 4T11C 006 24.

7. Literature

[1] Adamski M., Skowroński Z.: Interpretowane sieci Petriego – model formalny w zintegrowanym projektowaniu mikroprocesorowych systemów sprzętowo-programowych, PAK 2/3, 2003, 17-20.
[2] Ferrarini L.: An Incremntal Approach to Logic Controller Design with Petri Nets, IEEE Transactions on System, Man, and Cybernetics, Vol. 22, № 3, 1992, 461-474.
[3] Heiner M.: Petri Net Based System Analysis Without State Explosion, in Proceedings of. High Performance Computing'98, Boston, April 1998, 394-403.
[4] Janicki R., Koutny M.: Using Optimal Simulations to Reduce Reachability Graphs, in Proceedings of the 2nd International Conference CAV'90, 166-175, Springer 1991.
[5] Janicki R., Lauer P. E., Koutny M., Devillers R.: Concurrent and Maximally Concurrent Evolution of Non-Sequential Systems, Theoretical Computer Science, 43, 1986, 213-238.
[6] Karatkevich A.: Optimal Simulation of α-Nets, in: Proceedings of the Polish-German Symposium SRE'2000, Zielona Góra, 2000, 217-222.
[7] Karatkevich A., Zakrevskij A.: Analysis of Petri Nets by Means of Concurrent Simulation, in Proceedings of the International Conference PARELEC, Warsaw, September 2002, 87-91.
[8] Murata T.: Petri Nets: Properties, Analysis and Applications, Proceedings of IEEE, vol. 77, № 4, April 1989, pp. 548-580.
[9] Valmari A.: State of the Art Report: Stubborn Sets, Petri Net Newsletter, April 1994, 6-14.
[10] Wolper P., Godefroid P.: Partial-Order Methods for Temporal Verification, in Proceedings of the 4th International Conference CONCUR'93, 233-246, Springer 1993.
[11] Zakrevskij A., Karatkevich A., Adamski M.: A Method of Analysis of Operational Petri Nets, in Proceedings of the 8th International Conference ACS'2001, 449-460, Kluwer Academic Publishers, 2002.
[12] Закревский А. Д.: Параллельные алгоритмы логического управления, ИТК АНБ, 1999.

Artykuł recenzowany