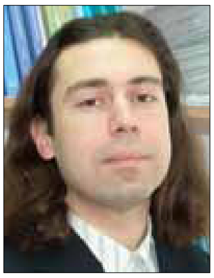
Andrei KARATKEVICH
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Formal Verification of FSM Networks

Dr inż. Andrzej Karatkiewicz



Dr inż. Andrzej Karatkiewicz ukończył w roku 1993 studia na Wydziale Techniki Obliczeniowej Mińskiego Radiotechnicznego Instytutu. Obronił pracę doktorską w Białoruskim Państwowym Uniwersytecie Informatyki i Radiotechniki w roku 1998. Od roku 2000 jest adiunktem w Instytucie Informatyki i Elektroniki na Uniwersytecie Zielonogórskim. Jego zainteresowania naukowe dotyczą głównie teorii sieci Petriego oraz zagadnień analizy i weryfikacji współbieżnych systemów sterowania.

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

Abstract

The paper presents some methods of detection of global and local deadlocks and some problems of more general kind in the control systems specified as concurrent automata. The proposed methods reduce the problems to solving the systems of logical equations. Modeling of the FSM networks by the Petri nets is used. The FSM networks are specified using the Statecharts. An example of a network analysis is given. The methods can be applied in the systems of computer-aided design of digital control systems as the part of verification process.

Keywords: digital controllers, FSM networks, Petri nets, Statecharts, parallelism.

Streszczenie

Artykuł przedstawia wybrane metody wykrywania globalnych i lokalnych zakleszczeń oraz niektórych szerszych problemów w układach sterowania, przedstawionych jako współbieżne automaty stanów. W proponowanych metodach wykrywanie problemów sprowadza się do rozwiązania układów równań logicznych. Stosuje się modelowanie sieci automatowych sieciami Petriego. W artykule przedstawiono także przykład analizy sieci automatowej. Proponowane metody mogą znaleźć zastosowanie w systemach komputerowego wspomagania projektowania sterowników cyfrowych na etapie weryfikacji.

Słowa kluczowe: sterowniki cyfrowe, sieci automatów, sieci Petriego, Statecharts, współbieżność.

1. Introduction

Finite State Machine (FSM) is the basic model of a discrete device, used in formal design of digital systems. However behavior of a complex system usually cannot be conveniently specified by single FSM because of concurrently acting objects and/or processes in such system. That is the reason of applying parallel extensions of finite automata in the formal design methods.

There are two main classes of such extensions: one is based on the Petri nets [1][10][13][15], another is based on parallel composition of the Finite State Machines [2][8]. The first one is convenient to use, when in the specified system the parallel processes divide and merge in a complex way, which cannot be described only by parallel and sequential composition of states. Petri nets provide more flexibility in the descriptions, but they are difficult to analyze. So, when the number of the parallel processes is constant or when parallel structure of a system can be described by the fork-join operations, where every "fork" has its "join", FSM networks are more convenient to use. However, it is just a question of convenience, because the modeling power of both kinds of parallel automata is the same.

Formal verification of FSM networks is simpler than such of the Petri nets, but there are also the non-trivial tasks caused by interaction between the automata in the networks, such as detection of deadlocks. In the paper some methods of deadlock and livelock detection are presented.

We use the event-based formalism of Statecharts [3] to describe the FSM networks. More concretely, we use a restricted, fully synthesizable model, representing a subset of Statecharts, as described in [8]. We consider only the asynchronous networks.

2. Networks without hierarchy: deadlocks…

For the flat (without hierarchy) FSM networks the number of active local states is the same in all global states, and every automaton at every moment has exactly one active state.

An internal event in an FSM network without hierarchy is absent because of a deadlock, only if all automata which can generate this event are deadlocked, and neither of them generates it in its deadlocked state as a static event. For static events it can be described as follows:

$$\varepsilon_k \vee \bigwedge_{e_k \in saction(p_i^j)} \bigvee_{p_l^j \in P^j, l \neq i} x_l^j = 1 \qquad (1)$$

For dynamic events:

$$\varepsilon_k \vee \bigwedge_{e_k \in taction(t_i^j)} \bigvee_{p_l^j \in P^j} x_l^j = 1 \qquad (2)$$

where:

ε_i - a Boolean variable meaning presence or absence of event e_i;
saction(p) – set of static events generated by state p;
taction(t) – set of dynamic events generated by transition p;
P^j – set of states of FSM number j;
x_i – a Boolean variable meaning activity or passivity of state p_i.

As far as in a deadlock, by definition, no transition can fire, for every active deadlocked state p the following condition holds:

$$\forall t\text{: } (out(t)=p) \Rightarrow (trigger(t) \cap Z \neq \varnothing) \qquad (3)$$

where:

out(t) – initial state of transition t;
trigger(t) – set of events necessary for transition t to be executed;
Z – set of internal events.

From (1) and (2) the following equations can be constructed for every state p_m which satisfies (3):

$$\overline{x_m} \vee \bigwedge_{out(t)=p_m} \bigvee_{e \in trigger(t)} \bigwedge_{e \in saction(p_i^j)} \bigvee_{p_l^j \in P^j, l \neq i} x_l^j = 1 \quad (4)$$

$$\overline{x_m} \vee \bigwedge_{out(t)=p_m} \bigvee_{e \in trigger(t)} \bigwedge_{e \in taction(t_i^j)} \bigvee_{p_l^j \in P^j} x_l^j = 1 \quad (5)$$

Equation (4) is to be used for the Moore automata, equation (5) – for the Mealy automata. Such equations for all states satisfying (3) together with the characteristic function (which usually can be calculated in a reasonable time [7][8]) specify a system of logical equations, which roots correspond to the reachable deadlocks.

3. …and livelocks

A more general class of behavioral problems than the deadlocks is unreachability of local states, which covers both deadlocks and livelocks (a livelock is a situation when two or more processes can change their states, but cannot attain some of the states). If some local states are inactive in all reachable global states, this can be directly detected from the characteristic function. However characteristic function does not answer the question, whether a local state, initially reachable, can become unreachable, when the system attains one of the global states.

Let us formulate the following task: an FSM network without hierarchy is given. Detect such global states of the network, from which some local states are not reachable, if such global states exist.

An FSM network can be modeled by a Petri net by applying the algorithm presented in [4]. A simple example of such modeling is shown in Fig. 1.

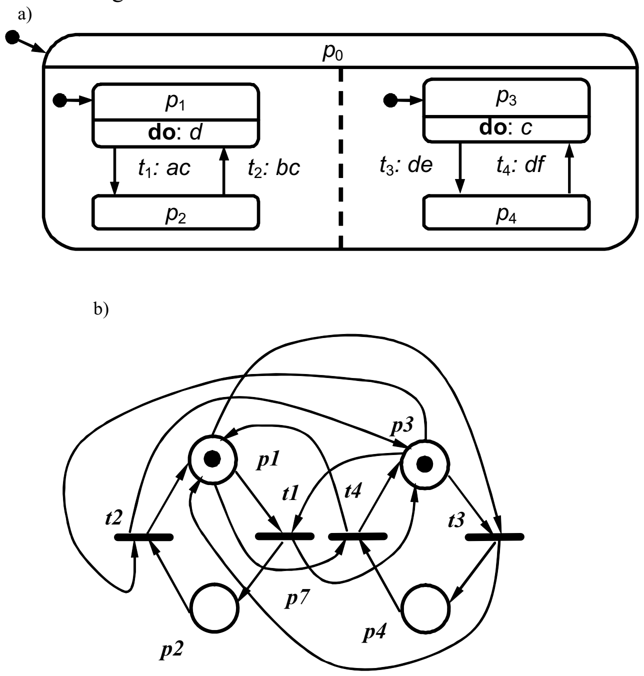


Fig. 1. An FSM network (a) and the modeling Petri net (b)
Rys. 1. Sieć FSM (a) i modelująca sieć Petriego (b)

Below we detect the unreachable states by analyzing the modeling Petri nets. We show that existing in such net a siphon of certain kind means, that some states may become unreachable.

But before analyzing a Petri net, it is reasonable to reduce it, if it is possible. A net may contain the sequential fragments without any synchronization with other parallel processes. Such a fragment corresponds to a part of state transition graph of an FSM, in which the automaton reacts only on the external events and does not generate the internal events, i.e. does not communicate with other automata of the network. Then the following operation should be performed with such subnet.

1 If the subnet has only one input place and one output place, and no place of it is initially marked or its input place is initially marked, replace the whole subnet by single place (if the input place is initially marked, the introduced place should be initially marked). Otherwise execute 2-4:
2 Check for every input place, which output places are reachable from it.
3 For every input place add a direct transition to every output place reachable from it.
4 Remove all other places and transitions of the subnet.

In Fig. 2 an example of such reduction is shown.

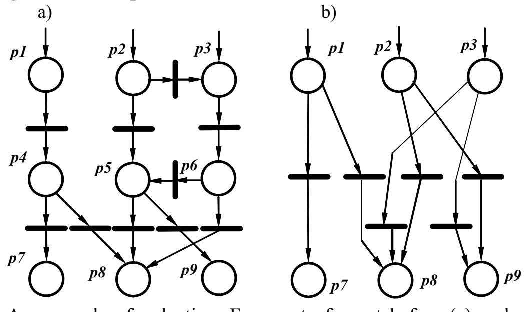


Fig. 2. An example of reduction. Fragment of a net before (a) and after (b) reduction
Rys. 2. Przykład redukcji. Fragment sieci przed redukcją (a) i po (b)

The proposed method is based on the following affirmation.

Affirmation 1. Let *N* be an FSM network and *Σ* - the Petri net modeling it. If there is siphon *D* in *Σ* such that there is no FSM in *N* for which *D* contains all places corresponding to it, then there exists a global state of *N* such that no local states corresponding to the places of *D* are reachable.

Proof. If *D* does not contain the places corresponding to all states of any FSM in the network, then there exists a global state *M'* of *N* such that no place in *D* is marked in the corresponding marking M of *Σ*. No marking which marks any place belonging to *D* is reachable from *M*, hence no such global state is reachable in *N* from *M'*, that a local state corresponding to a place in *D* is active.

Not all situations in which local states may become unreachable, can be captured by calculation of siphons, which is illustrated by Fig. 3. The Petri net modeling presented FSM network has no deadlocks of the kind mentioned in Affirmation 1, but one of the local states may become unreachable. However, the class of the situations which can be detected by means of calculation of siphons covers all global and local deadlocks and part of livelocks. For detailed proof of this statement see [6].

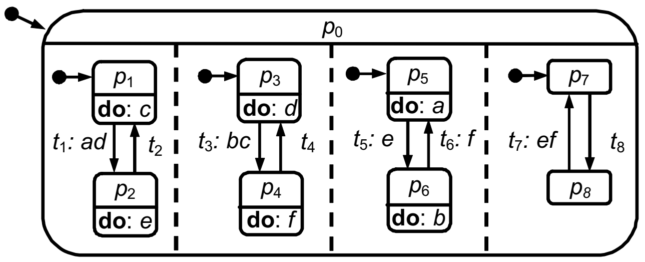


Fig. 3. State p_7 can become unreachable
Rys. 3. Stan p_7 może stać się nieosiągalny

There are lot of methods of calculation of siphons in the Petri nets. An efficient approach to such calculation is based on representation of the net structure by means of a system of Boolean equations, which roots correspond to the siphons [9][14][15]. Such equations can be solved in several different ways: by combinatorial search in the ternary matrices [14], by applying Thelen's prime implicant method [12] or Gentzen symbolic deduction [11]. Review of the methods of siphon detection and the bibliography can be found in [12].

Below, the algorithm detecting possible unreachability situations for given FSM network *N* is described.

Algorithm 1

1. Create for *N* the modeling Petri net *Σ*.
2. If possible, perform reduction of *Σ*, as described above.
3. Detect the siphons of *Σ*, satisfying the condition formulated in Affirmation 1.
4. Every obtained siphon *D* specifies a set of global states (all the global states in which no local states corresponding to places in *D* are active) and a set of local states (corresponding to the places belonging to *D*) not reachable from them. Check reachability of the specified global states from the initial state by means of the characteristic function.

4. Hierarchical networks

In flat FSM networks, activity of a local state in a deadlock implicates activity of some other local states, as it is described by equations (4,5). In hierarchical networks, lack of an event does not necessary mean that an FSM which can generate this event is deadlocked; it is possible that this FSM is not active at all, and another FSM at a higher hierarchy level is deadlocked. We propose the following approach for this case: consider for every local state satisfying (3), *passivity* of which local states it implicates [4][6]. The difference in comparison to the case of flat nets and equations (4,5) is that for the flat nets both local and global deadlocks can be detected; and the proposed method for the hierarchical Petri nets allows to detect only the global deadlocks. In a global deadlock, only the static and external events can be available, and for the static events we can write:

$$\varepsilon_k = \bigvee_{e_k \in saction(p_i)} x_i \qquad (6)$$

It follows, that a local state p_m can be active in a deadlock, only if

$$\overline{x_m} \vee \bigwedge_{out(t)=p_m} \bigvee_{e \in trigger(t)} \bigwedge_{e \in saction(p_i)} \overline{x_i} = 1 \qquad (7)$$

Solving of the system of equations consisting of equations (7) for every local state satisfying (3) together with the characteristic function, allows to obtain all reachable global deadlocks.

5. Example

Let us consider the network presented in Fig. 4. To simplify the diagram, only the internal events are shown.

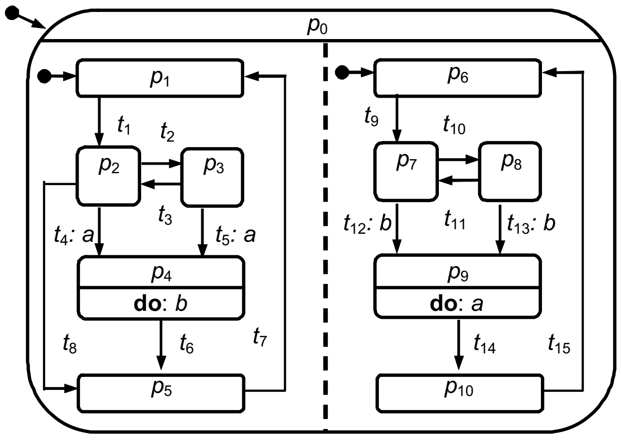


Fig. 4. An example of FSM network
Rys. 4. Przykład sieci automatów

Presented FSM is flat; it has no deadlocks, because there are no local states satisfying (3). Modeling Petri net for this example (after reduction) is shown in Fig. 5. All places and transitions of the net correspond to local states and transitions of the network shown in Fig. 4, excluding *MP1*, which is a "macroplace" corresponding states p_9 and p_{10} from Fig. 4.

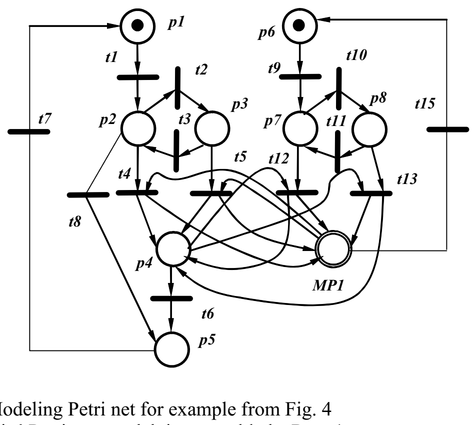


Fig. 5. Modeling Petri net for example from Fig. 4
Rys. 5. Sieć Petriego, modelująca przykład z Rys. 4

In this net the siphon {*p4*, *MP1*} can be detected, which satisfies the condition from Affirmation 1. That means, that from any global state, in which the first FSM is in one of the states p_1, p_2, p_3, p_5, and the second FSM is in one of the states p_6, p_7, p_8, no global state is reachable in which any of the states p_4, p_9, p_{10} is active. So, in this case a livelock is detected: neither of two automata can attain states p_4 and p_9, correspondingly, but at the same time neither of them can be deadlocked.

6. Summary

FSM networks are a simple and easy-to-understand formalism allowing to specify behavior of parallel logical controllers and of the discrete systems in general. It is used in many CAD systems. However, for a parallel system such as an FSM network, the complex verification tasks arise, which do not exist for the single FSMs and are related to checking correctness of interaction between the automata.

There are two main groups of verification tasks for the parallel control systems: quantitative (first of all, whether the system meets time requirements) and qualitative (first of all, whether the system can be deadlocked – as a whole or its parts). Verification via simulation is not enough here, because it cannot guarantee full covering of any class of problems. So, formal verification is important. In this paper we concentrate on the second group of tasks and describe briefly the main original results obtained in this area during last years.

For the one-level FSM networks we propose an analysis method, based on the simulation of the networks by Petri nets and finding in the modeling Petri nets the siphons of special kind. The method allows to detect all possibilities of global deadlocks and some possibilities of livelocks. Another proposed method, based on solving certain logical equations, detects all possible global and local deadlocks.

Next, we consider more general case of FSM networks, where. hierarchy is added. For such case we propose another logical algebraic method, which detects all possible deadlocks of the system and – with the help of characteristic function – checks, whether the detected deadlocks are indeed reachable from the initial state(s) of the system.

7. References

[1] Adamski M., Karatkevich A., Węgrzyn M. (editors): Design of Embedded Control Systems, Springer 2005.
[2] Gajski D.D., Vahid F., Narayan S., Gong J.: Specification and Design of Embedded Systems, Prentice-Hall 1994.
[3] Harel D.: Statecharts: a Visual Formalism for Complex Systems, Science of Computer Programming, № 8, 1987, 231-274.
[4] Karatkevich A.: Deadlock Analysis in Statecharts, Proceedings of FDL, September 2003, 414-424.
[5] Karatkevich A.: Detection of the Unreachable States in FSM Networks, Proceedings of the International Conference CAD DD, November 2004, 47-54.
[6] Karatkevich A.: Dynamic Analysis of Petri Net-Based Discrete Systems, Springer LNCIS vol. 356, 2007.
[7] Łabiak G.: Symbolic State Exploration of UML Statecharts for Hardware Description, Design of Embedded Control Systems, Springer 2005, 73-83.
[8] Łabiak G.: Wykorzystanie hierarchicznego modelu współbieżnego automatu w projektowaniu sterowników cyfrowych, Oficyna Wydaw. Uniwersytetu Zielonogórskiego, 2005.
[9] Minoux M., Barkaoui K.: Deadlocks and Traps in Petri Nets as Horn-Satisfability Solutions and Some Related Polynomially Solvable Problems, Discrete Mathematics, Vol. 29, 1990, 195-210.
[10] Murata T.: Petri Nets: Properties, Analysis and Applications, Proceedings of IEEE, Vol. 77, № 4, April 1989, 548-580.
[11] Tkacz J.: Wykrywanie zastojów w sterownikach logicznych metodą symbolicznego wnioskowania Gentzena, Pomiary Automatyka Kontrola, Czerwiec 2006, № 6bis, 11-13.
[12] Węgrzyn A., Karatkevich A., Bieganowski J.: Detection of Deadlocks and Traps in Petri Nets by Means of Thelen's Prime Implicant Method, Applied Mathematics and Computer Science, Vol. 14, № 1, 2004, 113-121.
[13] Zakrevskij A.D.: Parallel Autoimaton, Doklady AN BSSR, Vol. XXVIII, № 8, 1984, 717-719 (in Russian).
[14] Zakrevskij A.D.: To Checking Liveness of Ordinary Petri Nets, Doklady AN BSSR, Том XXIX, № 11, 1985, 1006-1009 (in Russian).
[15] Zakrevskij A.D.: Parallel Algorithms of Logical Control, Institute of Engineering Cybernetics of NANB, 1999 (in Russian).

Tytuł : Formalna weryfikacja sieci automatów skończonych.

Artykuł recenzowany