

Deadlock Detection in Discrete Concurrent Systems

Andrei Karatkevich

Andrei Karatkevich - University of Zielona Góra, ul. Podgórna 50, 65-246 Zielona Góra, POLAND.

Abstract

In this paper a method of deadlock detection in the concurrent discrete systems is given. Sequent automaton is used as a universal model of discrete systems. The method is based on constructing a sub-graph of reachability graph of the system. It can be considered as a generalization of the stubborn set method for Petri nets.

Keywords

– Discrete systems, state spaces, sequent automaton, concurrency, deadlocks, verification.

1. Introduction

The most known mathematical models used in design of digital electronic devices are Boolean functions (for combinational circuits) and finite state machines (FSM, for sequential circuits) [1]. A complex concurrent system (if it is synchronous) can be described by single FSM, but such possibility is rather theoretical, because of the so-called state explosion problem (state space of such system is a subset of the Cartesian product of the state spaces of its components). That's why the concurrent models are widely used in theory and practice of design of discrete devices. The most known of them are the Petri nets [2], FSM networks and numerous models and languages based on them, such as interpreted Petri nets [4] and Statecharts [5].

Verification of a concurrent system is a complex task, because interaction between the parts of such system may cause some problems which do not exist in sequential systems. One of them is deadlock. This term is used in computer science mainly in the next sense: "A condition that occurs when two processes are each waiting for the other to complete before proceeding" [6]. We use this word in the Petri net sense [2]: a state in which no transition can be executed. Deadlock detection is one of the important analysis problems of the concurrent systems.

There is a well-known method of deadlock detection in Petri nets, the so-called stubborn set method [7]. It is efficient for classical Petri nets, but its applicability to verification of the real systems is limited, because usually the interpreted Petri nets provide more adequate descriptions of the systems, and the stubborn set method cannot be directly applied to such nets (if they have internal variables).

In this paper we propose a method of deadlock detection, which is similar to the stubborn set method, but can be applied to a wider class of concurrent discrete systems,

including interpreted Petri nets and logical nets. The universal model we use is the sequent automaton [11]; an ordinary interpreted Petri net can be directly converted into such system.

II. Definitions

A Petri net [2] is a triple $\Sigma = (P, T, F)$, where P is a set of places, T – a set of transitions; $P \cap T = \emptyset$; $F \subseteq (P \times T) \cup (T \times P)$. For $t \in T$ $\bullet t$ denotes $\{p \in P | (p, t) \in F\}$; $t \bullet$ denotes $\{p \in P | (t, p) \in F\}$; $\bullet t$ and $t \bullet$ are the sets of input and output places, respectively; they never can be empty. The similar notation can be used for places ($\bullet p$ and $p \bullet$) and for sets of transitions or places.

A marking of a net is defined as a function $M: P \rightarrow \{0, 1, 2, \dots\}$. It can be considered as a number of tokens situated in the net places. Number of tokens in a place p for marking M is denoted as $M(p)$. Initial marking is denoted as M_0 .

A transition t is enabled and can fire (be executed) if $\forall p \in \bullet t: M(p) > 0$. Transition firing removes one token from each input place and adds one token to each output place of it. A marking that can be reached from M by a firing sequence is called reachable from M ; the set of reachable markings is denoted as $[M]$. A net is safe, if $\forall M \in [M], \forall p \in P: M(p) \leq 0$.

A transition is live if there is a reachable marking in which it is enabled; otherwise it is dead. A deadlock is such a marking that all the transitions are dead. Graphically Petri nets are presented as oriented bipartite graphs with two subsets of nodes corresponding to places and transitions and the arcs joining places to transitions or transitions to places. Tokens are shown as dots inside places (Fig. 1).

An interpreted Petri net [9,10] is (we use one of several known definitions) a Petri net such that a sequent s_i is associated to every transition t_i of it. A sequent [9] is an expression of kind $f_i \vdash k_i$, where f_i is a Boolean function, and k_i is an elementary conjunction. (Symbol \vdash denotes a cause-effect relation between the events.) An enabled transition of such net fires, if and only if in the corresponding sequent $f_i = 1$. Firing of the transition leads to assigning to all the variables appearing in k_i the values, for which $k_i = 1$. The Boolean variables appearing in the sequents are divided into 3 sets: X – input variables, appearing only in left parts of the sequents; Y – output variables, appearing only in the right parts; and Z – internal variables, appearing in both. Values of the input variables depend on the outer world; values to the

internal and output variables can be assigned only by the transition firing. An example of interpreted Petri net is shown in Fig. 2. Underlying net for it is the net from Fig. 1.

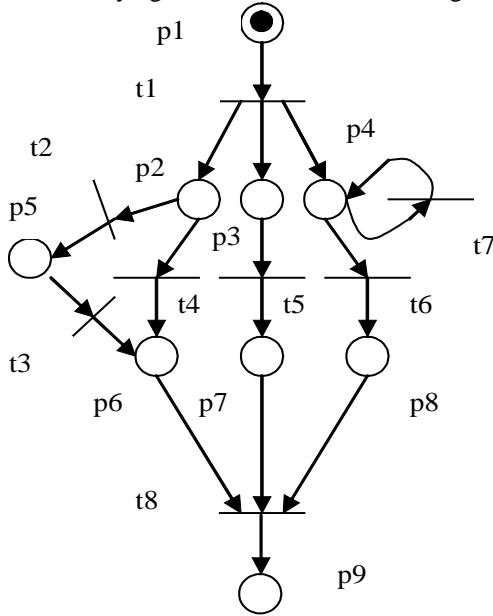


Fig.1 A Petri net.

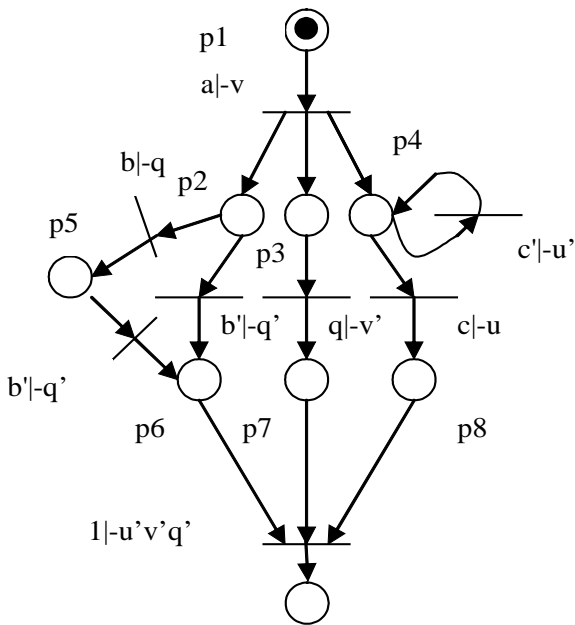


Fig.2 An interpreted Petri net.

A sequent automaton [9] is a system S of sequents $s_i = f_i | -k_i$. Set of the arguments of all the functions in the system is divided into 3 sets, as for the interpreted Petri nets.

The next interpretation is used: if at moment t the condition $f_i = 1$ (the sequent is enabled), then conjunction k_i will attain value 1 (after an arbitrary delay and if during that delay f_i has not changed its value); such change of the values of the corresponding variables is called sequent firing. Similarly to the Petri nets, sequents fire one by one, and for every firing all the variables' values change simultaneously. The internal and output variables keep their values, when no sequent firing changes them.

A safe Petri net, interpreted or not, can be easily described by a system of sequents. (1) corresponds to the net from Fig. 1, (2) – to the net from Fig. 2. Variables x_i correspond to places p_i . Negation of a variable x is denoted by x' .

$$\begin{aligned}
 x_1 &| -x_1'x_2x_3x_4 \\
 x_2 &| -x_2'x_5 \\
 x_2 &| -x_2'x_6 \\
 x_3 &| -x_3'x_7 \\
 x_4 &| -x_4'x_8 \\
 x_5 &| -x_5'x_6 \\
 x_6x_7x_8 &| -x_6'x_7'x_8'x_9
 \end{aligned} \tag{1}$$

$$\begin{aligned}
 x_1a &| -x_1'x_2x_3x_4v \\
 x_2b &| -x_2'x_5q \\
 x_2b' &| -x_2'x_6q' \\
 x_3q &| -x_3'x_7v' \\
 x_4c &| -x_4'x_8u \\
 x_4c' &| -u' \\
 x_5b' &| -x_5'x_6q' \\
 x_6x_7x_8 &| -x_6'x_7'x_8'x_9u'v'q'
 \end{aligned} \tag{2}$$

2. Stubborn Set Method for Petri Nets

A set TS of the transitions of a Petri net at marking M is a stubborn set, if a) every disabled transition in TS has an empty input place p such that all transitions in $\bullet p$ are in TS ; b) no enabled transition in TS has a common input place with any transition (including disabled ones) outside TS ; and c) TS contains an enabled transition [7].

The basic stubborn set method builds a reduced reachability graph (RRG) in the next way: for every considered state TS is calculated, and only firing of enabled transitions belonging to TS is simulated.

The next statement describes the fundamental property of the method [12]: RRG contains all deadlock states of the system

that are reachable from the initial states. All deadlock states of the RRG are deadlock states of the system.

The next example illustrates the method. Fig. 3 shows the complete reachability graph of the net from Fig. 1. Fig. 4 shows its RRG (one of the possible ones), created with the stubborn set method. The RRG contains one deadlocks, hence the Petri net has one reachable deadlock.

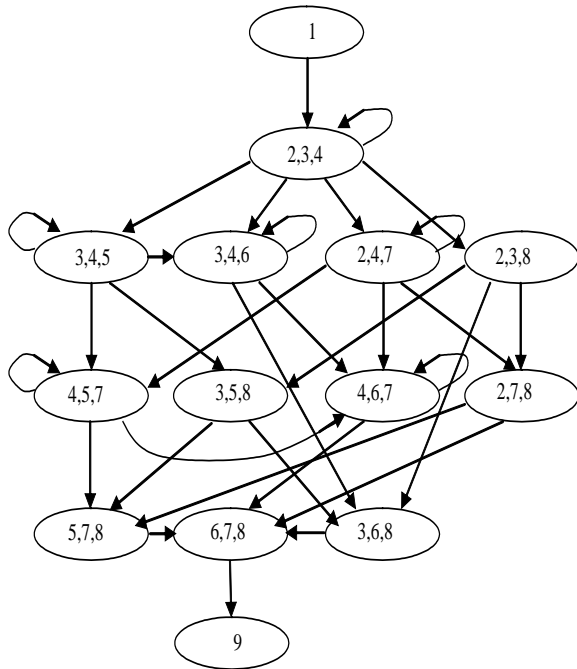


Fig.3 Reachability graph for net from Fig. 1.

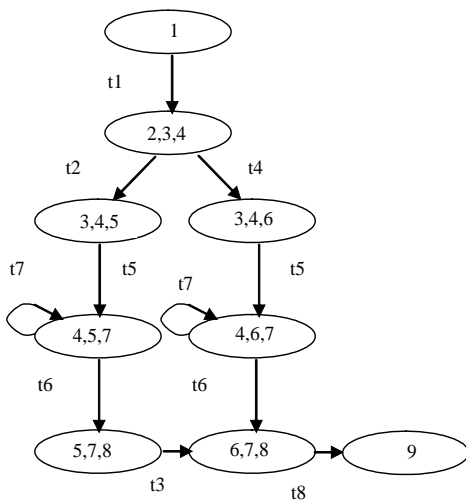


Fig.4 RRG for the net from Fig.1, constructed with the stubborn set method.

3. Limitations of the method

It is easy to see, that the method does not work properly for the interpreted Petri nets with internal variables, because it does not take into account the interaction between concurrent branches of the net by means of those variables. For example, Petri net from Fig. 1 has one deadlock, which can be detected by the stubborn set method. The interpreted Petri net form Fig. 2, having the same structure, has two reachable deadlocks – if the initial value of q is 0, and when place p_2 has a token, $b=0$, then a token cannot leave place p_3 , because the condition $q=1$ of firing of t_5 is never satisfied. So, marking in which places p_3, p_6, p_8 have tokens, may be a deadlock.

Stubborn set method, applied to the underlying net, cannot detect it. If during the RRG constructing the internal variables are taken into account, but the stubborn sets are constructed according to the basic method, the results may be interesting, but still unsatisfactory.

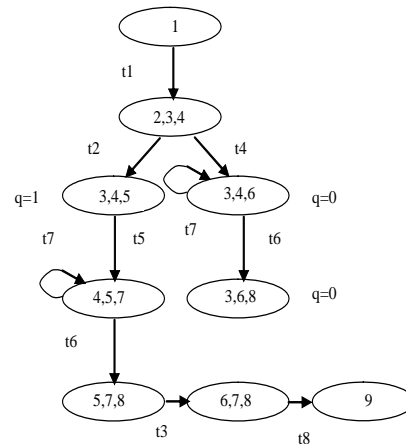


Fig.5 RRG for the net from Fig.2 (variant 1)

Applying the stubborn set method to a non-interpreted Petri net, we often have more than one variant of stubborn set for given marking. Size of RRG may remarkably depend on those choices, but in any case the set of detected deadlocks will be the same [7]. If we apply the method to an interpreted net, the deadlocks may be detected or not, dependently of the selected stubborn sets. For our example, one of the variants is shown in Fig. 5, another in Fig. 6. The first variant shows that two deadlocks are possible. However, the second variant shows only one deadlock, the other one remains undetected. The example shows, that the stubborn set method “as it is” does not work for interpreted Petri nets. It should be extended to take into account interaction via internal variables in such net. And if an algorithm allowing detecting deadlocks in interpreted Petri nets will be designed, it will work also for sequent automata, because a sequent automaton can be described as a form of interpreted Petri net (and vice versa). So, it makes sense to concentrate on deadlock detection for

sequent automata; in such a way a deadlock detection method for the interpreted Petri nets will be obtained too.

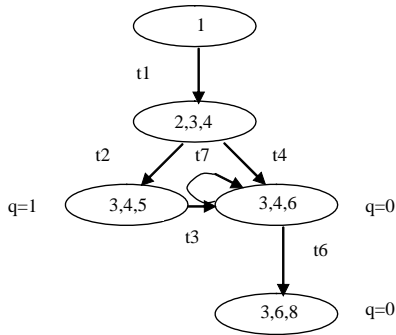


Fig.6 RRG for the net from Fig.2 (variant 2)

4. Some intuitions

It is worth noting, that the stubborn set method uses the fact that function of a transition enabling is monotonous. There are in general case several conditions of transition enabling (represented by input places), and “adding” something (a token) can satisfy an unsatisfied condition, but cannot break a condition already satisfied. For a sequent automaton equivalent to a safe Petri net, all the Boolean functions in the left parts of the sequents will be the elementary conjunctions of the Boolean variables without negations, i.e. they are monotonous (for any two input values σ and τ such that $\sigma \leq \tau$: $f(\sigma) \leq f(\tau)$).

The first condition of a stubborn set (see Section II) requires, that if a transition belonging to it is disabled, then for any unsatisfied condition all the possibilities of it to be satisfied are checked. The second condition requires, that if firing of a transition $t \in T_S$ can destroy a possibility of transition t' firing (even if it is disabled that moment), then t' should belong to the stubborn set. So, maybe it is possible to formulate conditions for the systems of sequents in similar terms and to define on that base stubborn set for the sequent automata?

Let us try to do that. In general case a function in the left part of a sequent is not necessary monotonous or even linear (it is linear only for a simple sequent; a general-case sequent automaton can be transformed into equivalent simple sequent automaton [9], but it does not solve our problem). How can it be presented as a set of condition to be satisfied? The answer is evident – it should be converted into a product of sums form [3]. Every sum term of a POS specifies a necessary condition of the function to obtain value 1.

Now, how can assigning to some Boolean variables “reduce possibilities” of a Boolean function to have value 1? In this

case it is natural to consider a sum of products form of the function. Every product term of a SOP specifies a sufficient condition of the function to obtain value 1, so changing values of the variables in such a way, that it turns any of those products into 0, may (for some combinations of values of other arguments) turn the whole function from 1 to 0.

Of course, such reasoning is not of much help here in the case of standard SOP and POS forms. It makes sense to use the compact forms, at least the forms consisting of only prime implicants and prime implicants [8], correspondingly.

5. Stubborn Set Method for sequent automata

Basing on the section III, we propose the next method of deadlock detection for sequent automata.

1. For every function f_i obtain its SOP and POS forms, such

that every product term k_i^j is a prime implicant, and every sum term d_i^j is a prime implicant, correspondingly.

2. Generate a reduced reachability graph of the automaton in the next way. For every state under consideration, simulate only the firing of those sequents, which are enabled (or can become enabled for some combination of values of input variables; below the term “enabled” is used in this extended sense) and belong to a stubborn set SS . Set SS satisfies the next conditions: a) for every sequent $s_i \in SS$ such that

$f_i = 0$: there is $d_i^j = 0$ such that for every literal x appearing in d_i^j every sequent s_l such that x appears in k_l belongs to

SS ; b) for every enabled sequent $s_i \in SS$: every sequent s_m such that $\exists k_m^j : k_m^j \wedge k_i \equiv 0$ belongs to SS ; c) an enabled sequent belongs to SS .

Let us apply the method to the sequent automaton (2), which is equivalent to the interpreted Petri net from Fig. 2. Fig. 7 shows the RRG for it (states are shown by enumeration of the internal variables having value 1). It is one of several possible RRGs, but every RRG created with the algorithm described above contains the same deadlocks.

V. Conclusion

In this paper a method of deadlock detection for a concurrent digital system is presented. A general model known as sequent automaton is used, which allows modeling of wide class of discrete systems.

The method still requires theoretical and experimental verification, however the performed experiments demonstrate that it works properly.

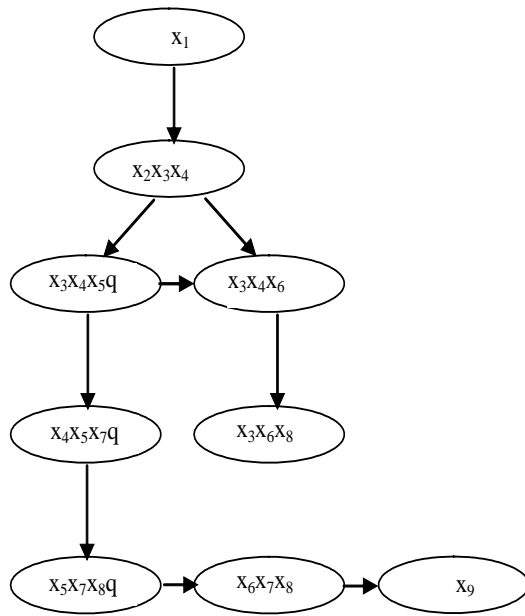


Fig.7 RRG for the sequent automaton (2).

References

- [1] В.М. Глушков, Синтез цифровых автоматов. М.: Физматгиз, 1962.
- [2] T. Murata, "Petri Nets: Properties, Analysis and Applications", Proceedings of the IEEE, vol. 77, pp.548-580, Apr. 1989.
- [3] R.S. Sandige, Modern Digital Design. McGraw-Hill series in electrical engineering, McGraw-Hill, USA, 1990.
- [4] R. David, H. Alla, Petri Nets & Grafcet Tools for modeling discrete event systems. N.Y.: Prentice-Hall, 1992.
- [5] D. Harel, "Statecharts: a visual formalism for complex systems", Science of Computer Programming, vol. 8, 1987.
- [6] www.webopedia.com
- [7] A. Valmari, "State of art report: Stubborn sets", Petri Nets Newsletter, 46, pp.6-14, 1994.
- [8] К.Г. Самофалов., А.М. Романкевич, В.Н. Валуйский, Ю.С.Каневский, М.М. Пиневиц, Прикладная теория цифровых автоматов. Киев: "Вища Школа", 1987.
- [9] А.Д. Закревский, Параллельные алгоритмы логического управления. Минск: ИТК НАН Беларуси, 1999.
- [10] G. Andrzejewski, Programowy model interpretowanej sieci Petriego dla potrzeb projektowania mikrosystemów cyfrowych. Zielona Góra: Uniwersytet Zielonogórski, 2003.
- [11] A. Zakrevskij, B. Steinbach, "Sequent automaton - a model for logical control", in Proceedings of the International Workshop "Discrete Optimization Methods in Scheduling and Computer-Aided design", Inst. of Eng. Cybernetics, Minsk, Belarus, Sept. 2000, pp. 205-209.
- [12] A. Valmari, "The State Explosion Problem", LNCS, 1491, pp.429-528, 1998.