Abstract: Deadlock detection is one of the important problems of analysis of discrete parallel systems. There is an efficient method of deadlock detection in Petri nets known as the stubborn set method. It cannot be directly used for the interpreted Petri nets; such net and its underlying Petri net may have different sets of deadlocks. In the article a method of deadlock detection for the generalized parallel discrete systems is presented with the theoretical results proving its correctness; concretization of the method for the interpreted Petri nets and sequent automata is proposed.

Keywords: Petri nets, formal models, simulation, verification

1. INTRODUCTION

Petri nets (Murata, 1989) are one of the basic models of parallel discrete systems. They specify a convenient formalism for high-level description of parallel algorithms, hardware, communication protocols, control systems and so on. A lot of analysis methods have been developed for Petri nets. One of the main analysis approaches is based on search of state space. This approach has to deal with the state explosion problem; so there are methods of analysis reducing the search to a relatively small part of state space (Heiner, 1998). Among them the persistent set approach (Wolper and Godefroid, 1993; Varpaaniemi, 1998) and the most elaborate technique in the family of persistent set methods known as the stubborn set method (Valmari, 1989; Valmari, 1994) are widely used.

But possibilities of application of the methods to analysis and verification of the real-life systems are limited, because behavior of such systems is affected by the factors which cannot be easily modelled by classical Petri nets. For example, there are several languages for specification of parallel logical control algorithms, based on the interpreted Petri nets (Zakrevskij, 1999; Andrzejewski, 2003; Lewis, 1995). Interpreted nets are a practical tool of control algorithms specification, but there can be an interaction between the parallel branches not at the level of underlying Petri net. So, behavioral properties of an interpreted net and its underlying Petri net can differ; also the sets of reachable deadlocks can differ. So developing of analysis methods for the interpreted Petri nets is practically important. One of such methods is based on modelling the interpreted Petri nets by the behaviorally equivalent classical Petri nets (Zakrevskij, 1999), but the obtained nets turn to be very complicated. It looks reasonable to develop methods allowing direct analysis of interpreted Petri nets.

In this paper we concentrate on the task of deadlock detection. In (Karatkevich, 2005) an attempt to formulate a stubborn set method for the interpreted Petri nets has been made, but the theoretical base was not satisfying. Here we present the generalization of the stubborn set method with the necessary theoretical base and its application to the interpreted Petri nets.
2. PRELIMINARIES

2.1 Parallel discrete systems

By a parallel discrete system we mean a system which state is described by a vector of discrete variables $M$ and can be changed by firing (execution) of the transitions; every transition has a necessary condition of firing being a binary function defined on $M$ (when it is satisfied, the transition is enabled). A transition firing changes values of some elements of $M$. Several transitions can be enabled in the same state $M$.

All the models described below are the concretization of this general model.

2.2 Interpreted Petri nets

We omit the well-known definitions of Petri nets and the related notions; see (Murata, 1989).

An interpreted Petri net (Zakrevskij, 1999; Andrzejewski, 2003) is a Petri net such that a sequent $s_i$ is associated to every transition $t_i$ of it. A sequent (Zakrevskij, 1999) is an expression of kind $f_i \vdash k_i$, where $f_i$ is a Boolean function, and $k_i$ is an elementary conjunction. An enabled transition of such net fires, if and only if in the corresponding sequent $f_i = 1$. Firing of the transition assigns 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. The underlying net for it is shown in Fig. 1.

2.3 Sequent automata

A sequent automaton (Zakrevskij, 1999; Zakrevskij, 2005) is a system $S$ of sequents $s_i = f_i \vdash k_i$. 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 some moment $f_i = 1$ (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 firing of the sequent. Like the transitions of 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 simple sequent automaton is a system $S$ of simple sequents, expressions $f_i \vdash k_i$ where both $f_i$ and $k_i$ are elementary conjunctions of Boolean variables (Zakrevskij, 2005).

2.4 Persistent set approach

The approach is based on calculation of a subset of successors from a state. Informally, a persistent set is a subset $T_P$ of transitions such that no transition firings outside $T_P$ affect $T_P$. The formal definition is given below, but preliminarily we have to define the notion of independent transitions of a parallel discrete system (Wolper and Godefroid, 1993; Varpaaniemi, 1998).
Definition 1. Transitions \( t_1, t_2 \) are independent at a state \( M \), if the following two conditions hold:

1. if \( t_1 \) (\( t_2 \)) is enabled in \( M \) and \( Mt_1M' \) (\( Mt_2M' \)), then \( t_2 \) (\( t_1 \)) is enabled in \( M \), if and only if \( t_2 \) (\( t_1 \)) is enabled in \( M' \) (independent transitions can neither disable nor enable each other);
2. if \( t_1 \) and \( t_2 \) are enabled in \( M \), then there is a unique state \( M' \) such that \( Mt_1t_2M' \) and \( Mt_2t_1M' \) (commutativity of enabled independent transitions).

Transitions \( t_1 \) and \( t_2 \) are globally independent, if they are independent at all reachable states. If the conditions do not hold, the transitions are said to be dependent.

For the classical Petri nets and for all the systems for which the diamond rule holds, the second condition is redundant. The diamond rule is a property, which can be formulated as follows: if \( Mt_1t_2M' \) and \( Mt_2t_1M'' \), then \( M' = M'' \). This property does not hold for some other parallel models, such as interpreted Petri nets and sequent automata.

Definition 2. A set \( T_P \) of transitions enabled in a state \( M \) of a parallel discrete system is persistent if and only if, for any sequence in the full reachability graph \( Mt_1t_2...t_nM_{n'} \) such that \( t_i \notin T_P \) and \( t' \notin T_P \) transition \( t' \) is independent with respect to all transitions in \( T_P \) at \( M_n \) (Varpaaniemi, 1998).

Let a persistent-set selective search be a selective search of the full reachability graph which, at each state \( S \) that it reaches, selects a nonempty set of enabled transitions \( T_P \) that is persistent in \( S \). Then the next theorem holds (Wolper and Godefroid, 1993).

Theorem 1. Let \( M \) be a state reached in a persistent-set selective search, and let \( D \) be a deadlock state. If \( D \) is reachable from \( M \), then \( D \) will also be reached by the persistent-set selective search.

Definition 3. A set \( T_S \) of the transitions of a Petri net at marking \( M \) is a stubborn set, if (1) every disabled transition in \( T_S \) has an empty input place \( p \) such that all transitions in \( \bullet p \) are in \( T_S \); (2) no enabled transition in \( T_S \) has a common input place with any transition (including disabled ones) outside \( T_S \); and (3) \( T_S \) contains an enabled transition (Valmari, 1989).

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

The next statement describes the fundamental property of the stubborn set method (Valmari, 1998).

Theorem 2. RRG contains all deadlocks of the system that are reachable from the initial states. Furthermore, all deadlocks of the RRG are deadlocks of the system.

Reduced reachability graph can be remarkably smaller than the full reachability graph; for example, full reachability graph for the net shown in Fig. 1 has 14 nodes and 32 arcs, and RRG only 9 nodes and 11 arcs.

Relation between stubborn sets and persistent sets is described by the next lemma (Varpaaniemi, 1998).

Lemma 1. The set of enabled transitions of a stubborn set is a nonempty persistent set.

3. A GENERALIZATION OF STUBBORN SET METHOD

Notion of independent transition is important for the persistent set methods. But, as it can be seen in Definition 3, it is essential to distinguish between the cases when one of dependent transitions can disable another and when one of them can enable another. For the systems for which the diamond rule does not hold it is important also to consider the kind of dependence which breaks the second condition of Definition 1.

The definition of stubborn sets can be re-formulated using different kinds of dependency between transitions. Below we formulate the definition for the generalized parallel systems.

Definition 4. A set \( T_S \) of the transitions of a parallel discrete system in state \( M \) is a stubborn set, if (1) for every disabled transition in \( T_S \) all the transitions that can enable do not hold in \( T_S \); (2) for every enabled transition in \( T_S \) all the transitions it can disable, all the transitions which can disable it, and all the transitions which are globally dependent in respect to it because of breaking the second condition of Definition 1 are in \( T_S \); and (3) \( T_S \) contains an enabled transition.

The next affirmations are the base of the stubborn set method for the generalized parallel systems.

Lemma 2. Let \( T_S \) be a stubborn set in a state \( M \) according to Definition 4. Then the set \( T_P \) of all the enabled transition in \( T_S \) is a persistent set.

Proof. According to Definition 2, it is enough to prove that for any sequence in the full reachability graph \( Mt_1t_2...t_{n'}M_{n'}t' \) such that \( t_i \notin T_P \) and

\[2\] Here by "\( t \) can enable (disable) \( t' \)" we mean "exists state, in which firing of \( t \) enables (disables) \( t' \)".
t' \notin T_P \text{ transition } t' \text{ is independent with respect to all transitions in } T_P \text{ at } M_n.

The proof proceeds by induction on } n. \text{ Let } n = 0 \text{ (then } M_0 = M). \text{ Every enabled transition outside } T_S \text{ is independent in respect of any enabled transition in } T_S \text{ by definition (condition (2) of Definition 4), and for } n = 0 \text{ the lemma holds.}

Now, assume that the lemma holds for every sequence of length } (n - 1) \geq 0 \text{ and let us prove that it holds for a sequence of length } n. \text{ Suppose transition } t' \text{ is dependent with respect to a transition } t \in T_P \text{ at } M_n. \text{ Then the next variants are possible: (1) } t' \text{ can disable } t; \text{ (2) } t' \text{ can be disabled by } t; \text{ (3) } t' \text{ and } t \text{ together break the second condition of Definition 1. In all three mentioned cases, according to Definition 4, } t' \in T_S. \text{ If } t_1 \text{ is disabled in } M, \text{ there is contradiction with assuming that } t_1 \notin T_P. \text{ If } t_1 \text{ is disabled in } M, \text{ then there is transition } t_2 \text{ (} 0 < i < n \text{) which enables } t'. \text{ According to Definition 4, } t_2 \in T_S. \text{ If } t_1 \text{ is enabled in } M, \text{ there is contradiction with the assumption that } t_1 \notin T_P.

**Theorem 3.** Reduced reachability graph of a parallel system, created in such way that in every considered state only firing of enabled transitions belonging to a set } T_S \text{ satisfying Definition 4 is simulated, contains all deadlocks of the system that are reachable from the initial states. Furthermore, all deadlocks of the RRG are deadlocks of the system.}

**Proof** of the first part follows directly from Lemma 2 and Theorem 1. The second part follows from the third condition of Definition 4.

Definition 4 is stronger than Definition 3; the first condition of Definition 4 in terms of Petri nets would look as follows: for every disabled transition in } T_S \text{ and its every input place } p \text{ all transitions in } \ast p \text{ are in } T_S. \text{ It means, that the stubborn sets for a Petri nets according to Definition 4 will be the same, as according to Definition 3, when the stubborn sets consist of the enabled transitions only, and can be bigger otherwise. Definition 4 can be, however, re-formulated in such a way, that applied to the Petri nets it would be equivalent to Definition 3. In this variant (let us call it Definition 4a) condition (1) looks as follows: for every disabled transition in } T_S \text{ and an unsatisfied necessary condition of its enabling all the transitions firing of which can satisfy this condition are in } T_S; \text{ the rest of conditions remain as in Definition 4. It is easy to see, that Lemma 2 and Theorem 3 hold for Definition 4a.}

![Fig. 3. An RRG for the net from Fig. 2, built with the basic stubborn set method](image)

**4. DEADLOCK DETECTION IN INTERPRETED PETRI NETS AND SEQUENT AUTOMATA**

Reachability graph } G_{int} \text{ of an interpreted Petri net } \Sigma_{int} \text{ is a subgraph of the reachability graph } G \text{ of the underlying net } \Sigma, \text{ because any interpretation can reduce possibilities of the net evolution and never expand them. RRG } G_R, \text{ constructed for } \Sigma, \text{ is a subgraph of } G, \text{ but not necessarily of } G_{int}, \text{ so } G_R \text{ can miss some information, important for analysis of } \Sigma_{int}.

Consider the next example. Petri net from Fig. 1 has one deadlock, which can be detected by the stubborn set method. The interpreted Petri net shown in Fig. 2 has two reachable deadlocks - if the initial value of } q \text{ is 0, and if } b = 0 \text{ when place } p_2 \text{ has a token, then a token cannot leave place } p_3, \text{ because the condition } q = 1 \text{ of firing of } t_5 \text{ is never satisfied. So, marking } \{p_3, p_6, p_8\} \text{ may be a deadlock. The stubborn set method, applied to the underlying net, cannot detect it.}

Applying the stubborn set method to a non-interpreted Petri net, we often have more than one variant of stubborn set for given marking, but in any case the set of detected deadlocks will be the same. If we directly 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 possible variants of RRG is shown in Fig. 3. It detects only one deadlock.

The example shows, that the stubborn set method "as is" does not work for interpreted Petri nets. The notion of stubborn sets has to be re-defined for such nets to take into account interaction via internal variables. And an algorithm allowing deadlock detection in interpreted Petri nets would work also for sequent automata, because a sequent automaton can be described as a form of interpreted Petri net (and vice versa, if the net is safe). 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.
Let us start from defining the notion of independent sequents, being a concretization of the notion of independent transitions (Definition 1).

**Definition 5.** Sequents $f_1 \vdash k_1$, $f_2 \vdash k_2$ are (globally) independent, if the following conditions hold:

1. no variable appearing in $k_1$ ($k_2$) is an argument of $f_2$ ($f_1$) (independent sequents can neither disable nor enable each other);
2. $k_1 k_2 \neq 0$ (commutativity of enabled independent sequents).

Definition 4a can be concretized for the sequent automata. To do it, we have to describe the case when a sequent can enable another one and when it can disable another one. It can be done by transforming an automaton into equivalent simple sequent automaton (Zakrevskij, 1999).

If all sequents are simple, then $s_1$ can enable $s_2$, if there is a literal $l_i$ occurring in both $k_1$ and $f_2$, and in the current state $l_i = 0$; $s_1$ can disable $s_2$, if there is variable $x$, occurring in $k_1$ with (without) negation and in $f_2$ without (with) negation. Every literal occurring in the left part of a sequent specifies a necessary condition of its enabling.

**Definition 6.** A set $S_S$ of the sequents of a simple sequent automaton at state $M$ is a stubborn set, if (1) for every sequent in $S_S$ there is a literal $l = 0$ in its left part (being an internal variable with or without negation) every sequent such that $l$ occurs in its right part belongs to $S_S$; (2) for every enabled sequent $s_i$ in $S_S$ every sequent $s_j$ such that $k_i f_j \equiv 0$ belongs to $S_S$; (3) for every enabled sequent $s_i$ in $S_S$ every sequent $s_j$ such that $k_i k_j \equiv 0$ belongs to $S_S$; (4) $S_S$ contains an enabled sequent $s_i$ such that $k_i = 0$ at $M$.

It follows from Theorem 3, that a reduced reachability graph of a simple sequent automaton built using the stubborn sets in the sense of the definition above contains all the reachable deadlocks of the automaton. And a safe Petri net, interpreted or not, can be easily described by a system of sequents (Karatkevich, 2005). For example, (1) corresponds to the net from Fig. 2. Variables $x_i$ correspond to places $p_i$.

\[
\begin{align*}
x_1 a &\vdash \overline{f_1 x_2 x_3 x_4 v} \\
x_2 b &\vdash \overline{f_2 x_5 q} \\
x_3 b &\vdash \overline{f_6 x_6 q} \\
x_3 q &\vdash \overline{f_3 x_7 \bar{v}} \\
x_4 c &\vdash \overline{f_4 x_8 v}
\end{align*}
\]

(1)

\[\text{or the sequent which can become enabled for some combination of values of input variables; below the term "enabled" is used in this extended sense.} \]

![Fig. 4. RRG for the interpreted Petri net from Fig. 2, built with Algorithm 1](image)

We propose the next method of deadlock detection for the interpreted Petri nets, based on the reasoning above. The algorithm constructs a reduced reachability graph for given safe interpreted Petri net $\Sigma_{int}$.

**Algorithm 1.**

1. Obtain the sequent automaton equivalent to the net $\Sigma_{int}$.
2. If the sequent automaton is not simple, transform it into an equivalent simple sequent automaton.
3. Remove all input variables from the sequents.
4. Generate a reduced reachability graph of the automaton in the next way. For every state under consideration, calculate a stubborn set $S_S$ satisfying Definition 6 and simulate only the firing of those sequents, which are enabled and belong to $S_S$.

Graph built with Algorithm 1 contains all the deadlocks of the net.

Let us apply the method to the interpreted Petri net from Fig. 2. Fig. 4 shows the RRG for it (inside the nodes the internal and output variables having value 1 in the corresponding states are shown). Both deadlocks are detected.

Of course, to detect the deadlocks in a sequent automaton (simple sequent automaton), Algorithm 1 without the first item (two first items) can be used.

5. CONCLUSION

The work introduces a general approach to analysis of parallel discrete systems and presents its
This paper is theoretical, and it is rather difficult at this stage to evaluate quantitative properties of the approach - a series of experiments with real systems would be necessary for that. Size of reduced reachability graph of a finite (bounded) parallel system, like size of full reachability graph, depends on the size of the system exponentially in the worst case. But the difference between full and reduced reachability graphs varies greatly for the different systems - in the best case size of the reduced reachability graph is linear, in the worst case it does not differ from the full reachability graph. Even for the classical stubborn set method for the Petri nets, however there were attempts to evaluate practically the efficiency of the stubborn set method (Varpaaniemi and Rauhamaa, 1992; Valmari et al., 1993), still the method has not been tried with some reasonably large examples, but we do not still yet have sufficient evidence to decide how good they really are. It seems now certain that they fail everywhere now and then, but it also seems that in the meantime they provide results varying from moderate to excellent. We may suppose that the results of experiments on the extended stubborn set method will be similar in this sense. Defining the class of the parallel systems which can be efficiently analyzed by the stubborn-set-like methods is a worthwhile direction of future work.

6. REFERENCES


