MODELLING DEADLOCK AVOIDANCE IN AGV SYSTEMS VIA COLOURED PETRI NETS

Michal Žarnay[1], **Ladislav Jančík**[2], **Petr Cenek**[1]

[1]University of Žilina, Faculty of Management Science and Informatics
01008 Žilina, Univerzitná 1, Slovak Republic
[2] Siemens Program and System Engineering s.r.o.
010 01 Žilina, J. M. Hurbana 21, Slovak Republic

michal.zarnay@fri.uniza.sk(Michal Žarnay)

Abstract

Deadlocks are undesirable states in resource allocation systems (RAS). Their avoiding presents a major issue in control of RAS. Out of extensive theory on this topic, we focus on comparison of two deadlock avoidance methods for sequential RAS: Banker's algorithm and C/D-RUN deadlock avoidance policy (DAP). We present their impact on a system of automated-guided vehicles (AGV system), modelled and analyzed by coloured Petri nets.

Keywords: Deadlock avoidance, Banker's algorithm, C/D-RUN DAP, coloured Petri net, AGV system.

Presenting Author's Biography

Michal Žarnay received the engineering degree in Information and Management Systems and the PhD. degree in Transportation and Communications Technology from the University of Žilina, Žilina, Slovak Republic, in 1999 and 2007, respectively. He works currently as an Assistant Professor in the Transportation Networks Department, Faculty of Management Science and Informatics, University of Žilina, where he is in charge of courses on Petri nets and management.
In 2008, he worked as researcher with the Institute for Traffic Safety and Automation Engineering at the Technical university of Braunschweig, Germany, in his main research area: modelling and analysis of systems control using Petri nets with applications to transportation and communication systems. In 2009, he joined the Group of Discrete Event Systems Engineering at the University of Zaragoza, Spain, to participate on theoretical research on decolourisation of coloured Petri net models and their further fluidification to obtain continuous Petri net systems.

1 Introduction

Deadlock state is a state of a system, where two or more system processes are blocked in their execution by waiting for resources that are occupied at the same time by the processes in the waiting list (example in Fig. 2). The waiting processes thus block and are blocked. Unblocking this state is possible only by an exceptional operation.

The deadlock states are undesirable in any system since they attack system effectiveness. They occur as a result of allocation of resources to processes in certain situations. Formalism used for analysis of this problem is called *Resource Allocation System* (RAS).

For the problem solving, a large variety of methods can be found in literature. We pick two of them for comparison: Banker's algorithm (BA) [1, 2] and C/D-RUN deadlock avoidance policy [3, 4]. The first one has been demonstrated to work on complex RAS with non-sequential processes with flexible routing and use of resources of multiple types at once [5, 6, 7]. The second one is a deadlock avoidance policy for a simpler system, sequential RAS with use of resources of multiple types at once, but without flexible routing of processes, and it is based on the formalism of Petri nets.

Both deadlock avoidance methods have been compared on an example of a system of automated guided vehicles (AGV) that is described e.g. in [8]. Model of the example has been implemented by means of coloured Petri nets [9] in CPN Tools [10].

In the following section, we introduce the sample AGV system used in the work. Sections 3 and 4 describe used formalisms for modelling and methods for deadlock avoiding. Section 5 provides some details about model construction, followed by results of our experiments and conclusion.

2 Example of AGV System

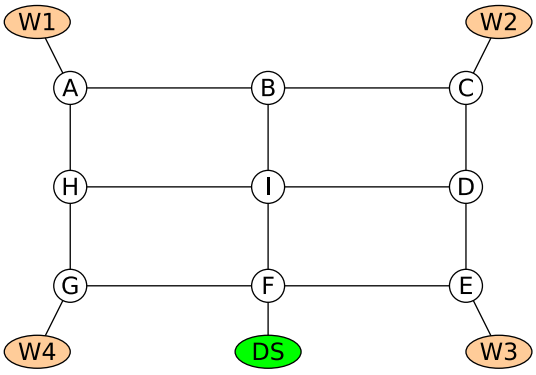

Fig. 1: AGV net used in example with docking station DS and 4 working stations Wi, $i = 1..4$.

As an example, we use the automated guided vehicle (AGV) system from [8] (network in Fig. 1) that can be used in modern factories or warehouses. The vehicles are stored in the docking station DS (its capacity is enough for all vehicles in the system) and they make trips to work stations Wi, $i = 1..4$ (every workstation accepts no more than one vehicle). A trip consisting of at least two work stations and starting and terminating in DS will be called mission. In our example, we use the following missions: $DS - W1 - W3 - DS$, $DS - W2 - W4 - DS$, $DS - W1 - W2 - DS$. Every network section can be occupied by at most one vehicle. For executing each mission, the vehicle is supposed to use a minimal number of network sections. If there are several routes available to reach the next destination, it selects one option arbitrarily. To keep the model simple, we accept a limitation that vehicles cannot be assigned new missions before coming back to docking station.

An example of a deadlock state in such a model is in Fig. 2.

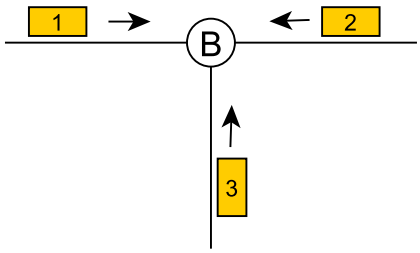

Fig. 2: Example of deadlock state in an AGV system: three vehicles approaching the intersection B occupy all network sections leading to it and wait for a section to be free to continue their mission.

3 Formalisms for Modelling

To compare the deadlock avoidance methods on a model of an AGV system, we have chosen resource allocation system implemented by means of hierarchical coloured Petri net.

Resource allocation system (RAS) is a system consisting of concurrently running processes that in certain stages, in order to get successfully completed, require exclusive use of certain number of system resources [11]. Resources are limited and re-usable as their allocation and de-allocation changes neither their character nor quantity. Main purpose of RAS is to solve problems arising from allocation of resources to processes. Based on properties of involved processes, a concrete RAS falls into a certain category [12] that decides on methods to be used for deadlock solving.

RAS for AGV system (denoted as AGV RAS) has vehicle missions as processes and network sections as resources. Vehicle has to traverse a sequence of network sections to fulfill a mission and to visit its defined points. Since a network section can be occupied by one vehicle at most, it is a resource allocated to the process exclusively. Because two network sections have at most

one common endpoint, they represent resources of different types. As a vehicle occupies only one section at a time (if it is not in DS), such an AGV RAS falls into the simplest category: sequential RAS of sequential processes with flexible routing and use of single resource unit at once by a process instance.

Petri net (PN) is a formalism used for modelling and analysis of systems with concurrent processes. It has graphical notation, precise mathematical language and analysis methods for specifying the system behaviour. Basic construction elements of PN are places, transitions, directed arcs and tokens. Places and transitions are two types of nodes in the net. Directed arcs link places with transitions, while no pair of nodes of the same type can be connected. Tokens are elements that move in the created network between places through arcs and transitions. Principal difference between places (drawn as circles) and transitions (rectangles) lies in their relation to tokens (black dots). Places can be marked with tokens. The number of tokens and their distribution in places represent the state of the net. Transitions take tokens from input places (i.e. places from which directed arcs lead to a transition) and send tokens to output places (i.e. places to which directed arcs lead from the transition). This process is called firing - it performs an action in the net, changing its state. In this way, it is also possible to change the overall number of tokens in the net. More details about Petri nets can be found in [13].

The basic PN formalism, called *Place/Transition* (P/T) Petri net, is often enriched or restricted to obtain enhancements or subclasses of PN. In this paper, we deal with hierarchical *coloured Petri net* (CPN), as it is introduced in [9]. Hierarchy allows dividing of a complex PN model into modules called sub-pages that are interconnected through special kind of nodes (substitution transitions, port, socket and fusion places). Colour is a symbolic name for value added to a token in the CPN, what distinguishes individual tokens (they are not black dots anymore). This requires additional specification for places, transitions and arcs, and thus allows construction of models with simpler net structure and added description, while keeping the same modelling power as would be with basic P/T PN.

CPN Tools [1] is a widely used tool for editing, simulation and analysis of hierarchical coloured Petri nets. Inscriptions are made in CPN ML, adjacent language to net structure in the CPN Tools. More on the tool can be found in [10].

For modelling of AGV RAS, the Petri net formalism is suitable to be used. A PN (non-coloured one) modelling AGV RAS has been called *System of Simple Sequential Processes with General Resource Requirements* (S^3PGR^2) in [14]. We extend this to the environment of CPN and use colours, when suitable.

In such a CPN, subnets of two types are found: process and resource subnets. A *process subnet* consists of places, transitions and arcs in a structure starting by an initial transition and ending with a final transition and describing causal relations between stages of a process. A stage (a task) in the process corresponds to a place and beginning and ending events of a task correspond to transitions. Together with a place for idle processes (let's denote it P_0), which connects the final transition with the initial transition of the process description, it makes a strongly connected component. The variants of flexible routing in the process description are created, when at least one place has at least two output transitions (conflict in Petri net, like in state machines) and another place has at least two input transitions, while all possible routes in the process subnet contain the place P_0. A *resource subnet* consists of one place and adjacent arcs. Content of the place represents actually free resources and arcs express their allocation and de-allocation to and from stages of processes. Typically, there are several process subnets, one for each modelled process, and one resource subnet in a RAS model.

For a description of the system's dynamic behaviour, we'll distinguish between process types and process instances. The *process type* is an abstract description of a process. The *process instance* is a concrete occurrence of a process according to a process type. In the CPN, the process type is modelled by the process subnet and the process instance by one or more tokens of one colour. A position of a token in a place of the process subnet represents a *stage* of the process. Similarly, there are resource types and resource instances. The former modelled by colours of a colour set for all resources (one resource type corresponds to one colour) and the latter by individual coloured tokens (number of tokens of a colour corresponds to number of resources of the respective resource type).

4 Solving Deadlock Problems

Deadlock problems in a system can be approached in one of three ways: deadlock detection and recovery, deadlock prevention and deadlock avoidance [11].

The *deadlock detection and recovery* approach lets the system run and if it accounters a deadlock state, a recovery procedure modifies the system back to a *safe state*, i.e. a state, from which all processes can finish their execution. It is suitable for systems with low probability of deadlock occurrence and low recovery cost.

The *deadlock prevention* approach aims at breaking at least one of four conditions of existence of deadlock, known as Conditions of Coffman [15]. It is suitable only for systems, where the necessary modification does not lower the system effectiveness in an unacceptable extent.

The *deadlock avoidance* approach uses information on current system state to decide whether it fulfills a requirement for a resource allocation or not. It is used in systems, where the previous two approaches are not acceptable. There is a large variety of methods for this approach, including the banker's algorithm and C/D-RUN deadlock avoidance policy.

4.1 Banker's Algorithm

The *banker's algorithm* (BA), first introduced in [1], uses information about a current system state to decide, whether a process allocation request can be fulfilled. It is called every time, when an allocation request is made. It checks, if the allocation leads to a *safe state*. If it does, the request can be fulfilled, otherwise, the requesting process must wait until another process returns resources. In order to decide about the state's safeness, the BA tries to order all active processes in such a sequence, so that each of the processes can be finished with resources that it currently occupies or that are currently available in the system or that are already returned from processes finished in the sequence prior to the tested process. If it succeeds in finding such a sequence, we say that the state is *ordered*, and since every ordered state is safe [2], the state is also safe. If it fails to find the sequence, we say that the state is *unordered*, which does not mean that the state is unsafe. However, the allocation request cannot be fulfilled. This is due to the suboptimality of the BA, while finding an optimal algorithm for solving the question about state safeness is a NP-hard problem.

Several modifications of banker's algorithm exist. We have chosen 3 modifications introduced in [2]: basic version A producing ordered states, version B producing partially ordered states and version C for V_1-ordered states.

4.2 C/D-RUN

The C/D-RUN deadlock avoidance policy (DAP) has been introduced in [2]. It is based on Petri net and its liveness property and structural subnet called siphon.

The C/D-RUN DAP adds a specific subnet in the Petri net model of the system that ensures avoiding states leading to deadlock states. Structure of the added places with initial tokens and their connection to existing transitions with arcs of given weight is calculated after solving a Mixed Integer Programming problem.

5 Model Construction

5.1 CPN Model of AGV RAS

The AGV RAS example has been implemented in the form of a hierarchical coloured Petri net model, where every mission is modelled in the form of process subnet. The subnets corresponding to three process types $P1$, $P2$ and $P3$ are visible on the top page in the hierarchy (Fig. 3). Each process type is further divided into three parts according to partial transfers among given destinations. Each of the partial transfers represented as substitution transition on the top page, is further defined on its sub-page. For instance, $P1$ is divided to transfers $DS - W1$, $W1 - W3$ and $W3 - DS$, i.e. three substitution transitions. There are eight sub-pages altogether, since one sub-page (DS_TO_W1) is used in two process types. That also indicates advantage of the hierarchical approach in further extension of the AGV RAS model, where partial transfers already defined in sub-pages can be easily reused in new process subnets on the top page.

Sub-pages specify vehicle movements by allocation and de-allocation of network sections. Position of a vehicle in a section is represented by a token in a place, a motion to another section by firing a transition - section notation used in place and transition names reflects direction of movement in the section (e.g. IB and BI are opposite movements in the same section BI). In case of sections adjacent to workstations, their notation is node-workstation-node (e.g. $AW1A$) to remark that a vehicle occupying such a net section keeps it reserved for coming back to the network after performing its loading/unloading job in workstation, since the workstation capacity is limited to one vehicle. Structure of the sub-page net is given by variant routes between nodes. For instance, the transfer $DS - W1$ is further defined on the sub-page DS_TO_W1 (Fig. 4) with 3 possible routes between nodes DS and $W1$, branching off to two directions after reaching the node F (place DSF) or after reaching the node I (place FI).

System resources are modelled in the place named *EDGE_RESOURCE* of the colour set $cEdge$, where every network section (edge), considered as a single instance of a single resource type, is represented by one value of a unique colour. Notation for sections is denoted by its nodes in alphabetical order (e.g. AB, BI or $CW2$). The tokens are delivered to the processes according to allocation and de-allocation of the sections via transitions in the execution of a vehicle mission.

5.2 Implementation of Banker's Algorithm

BA has been implemented by means of CPN ML, language available in the CPN Tools. Detailed description of its basic version A is to be found in [7]. Versions B and C have been implemented in an analogical way. This version of BA does not follow the algorithm defined specifically for AGV model in [8].

To include the BA in the model, there are two steps to carry out. At first, two places have been created: $PROCESS\_STATES$ keeps information about current state of the system in the form of data structure used by the BA and $BANKER\_ALG\_DATA$ information about allocated, currently available and later requested resources. Both places are connected to every transition, where an allocation or de-allocation operation is made. An example for the transition $IB{-}{>}BA$ in a net extract from the CPN model is in Fig. 5.

Secondly, the BA functions are called at every transition with an allocation request via its transition guard: the BA evaluation of the system state creates another condition for making the transition available and allowing the request to be fulfilled.

5.3 Implementation of C/D-RUN Policy

To include the C/D-RUN policy in the CPN model, a new place called $CD - RUN\_RESOURCE$ is added (Fig. 6). Its initial marking has been calculated according to the procedure described in [3, 4] – for the optimal bound, it contains the same initial marking as the $EDGE\_RESOURCE$ place with addition of one to-

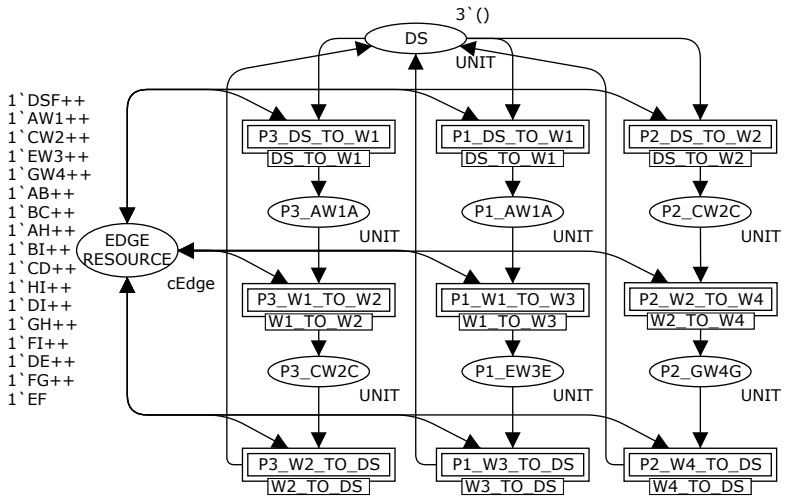

Fig. 3: Top page of the model.

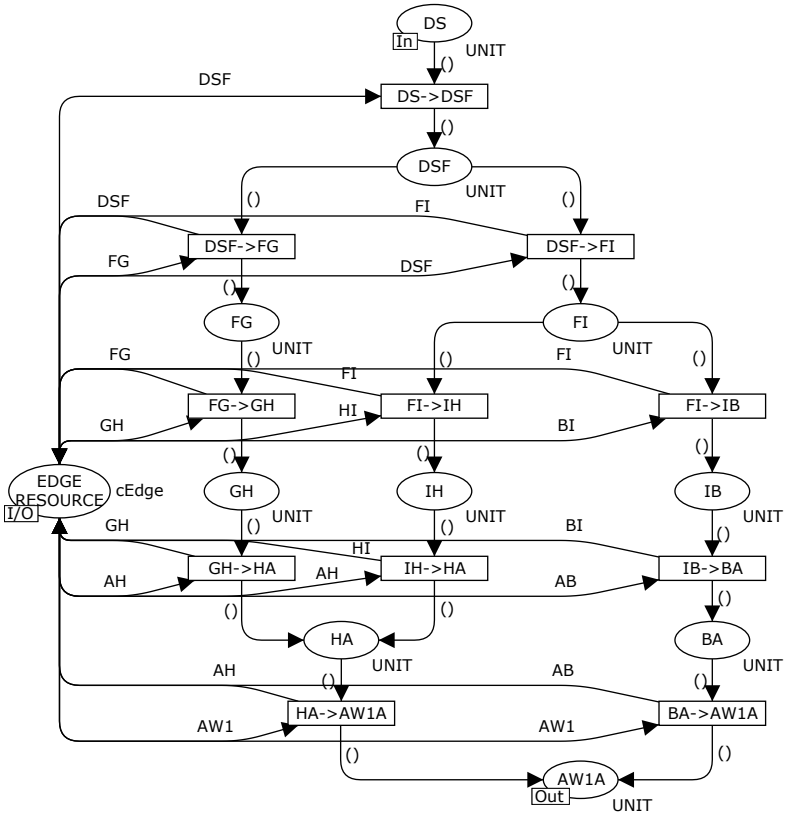

Fig. 4: Example of a subpage: vehicle transfer from docking station (DS) to workstation 1 (W1).

ken of colour AB.

Further details about construction of the model, implementation of both methods and analysis of the system can be found in [16].

6 Results

The presented model has been run with both methods of deadlock avoidance.

As the results in Tab. 1 show, the banker's algorithm restricts the state space of the controlled model much less than the C/D-RUN DAP.

In addition for 4 and more vehicles, the analysis results of the model with the C/D-RUN DAP have been the same as the results for 3 vehicles in the system, while all additional vehicles have been residing in the docking station during the whole model development. It means that the method does not allow movement of more than

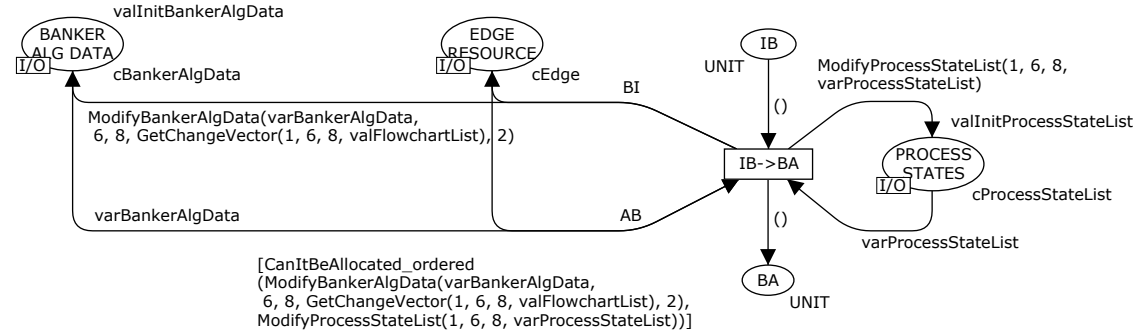

Fig. 5: Extract of a subpage: connection of Banker's Algorithm to the CPN model.

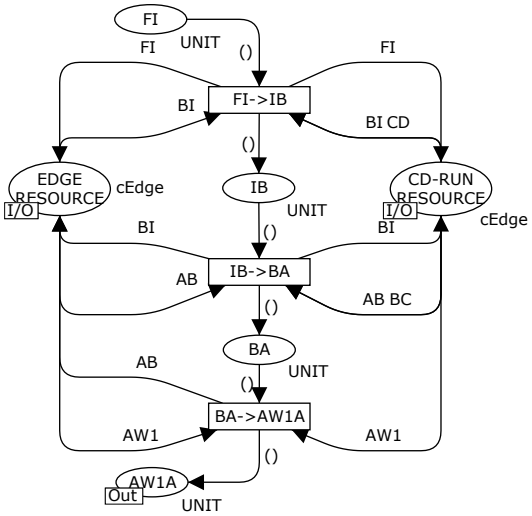

Fig. 6: Extract of a subpage: implementation of CD-Run.

3 vehicles in the modelled AGV system at the same time.

State space of the model with banker's algorithm with initial number of 4 vehicles or more can't be calculated because of restrictions of the CPN Tools. Instead of that a series of 25 simulation replications of 1000 steps have been done for each version of the algorithm. None of the replications finished in a deadlock state and the maximal number of active vehicles in one moment has been 7 for the BA algorithms A and B and 10 for the BA algorithm C.

Tab. 1: Comparing the methods in restriction of state space (percentage of states from original state space) for initial marking of 2 and 3 processes.

Method	Ordered states in %	
	2 processes	3 processes
C/D-RUN DAP	52	12
BA / algorithm A or B	97	91
BA / algorithm C	99	98

However, calculation of the state space, and thus calculation in one step, when deciding on fulfillment of an allocation request, takes more time in the model with banker's algorithms (see Tab. 2).

Tab. 2: Comparing computation time of state space for initial marking of 3 vehicles.

Method	No. of states in SS	Overall time [s]	Time per 1 state [ms]
C/D-RUN	5012	18	3.6
BA / A	39469	2799	70.9
BA / B	39469	2815	71.3
BA / C	42488	3444	81.1

7 Conclusion

Main contribution of this paper is in description of a coloured Petri net model of resource allocation system based on a sample AGV system and in the comparison of effectiveness of the C/D-RUN deadlock avoidance policy (DAP) to the banker's algorithm (BA). It shows that the C/D-RUN DAP restricts the state space of the system much more than the BA, but thanks to its nature based on Petri net, it is about 20 times faster. However, providing the calculation time of banker's algorithm in solving one allocation request in larger AGV systems is acceptable, it is highly advisable to prefer this method in their control applications over C/D-RUN DAP based on Petri net.

The formalism of hierarchical coloured Petri nets has shown to be suitable for performing the outlined task. Petri nets natively provide the necessary state space analysis and simulation. At the same time, construction of Petri net model with help of small number of building elements is rather easy and smooth (especially for experienced users), what is even reinforced by use of colours and hierarchy in the chosen subclass of high-level Petri nets. The environment of the CPN Tools, in addition, allowed implementing the BA and integrating it to the CPN model – thanks to the CPN ML. Flexibility of the approach has been utilized also by definition of test configurations, where changes in model parameters have been done easily and fast.

Result of this work can be considered in any application

area considering deadlock avoidance policies in control of simulation models or real systems. Our future work on this topic will focus on implementation of the BA in a specialized application for simulation of large logistic nodes called Villon [17].

8 References

[1] E. W. Dijkstra. Co-operating sequential processes. In F. Genuys, editor, *Programming Languages*, page 43112, New York, 1968. Academic Press. Reprinted from: Technical Report EWD-123, Technological University, Eindhoven, the Netherlands, 1965.

[2] M. A. Lawley, S. A. Reveliotis, and P. M. Ferreira. The application and evaluation of Banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 10:73–100, 1998.

[3] S. A. Reveliotis, M. A. Lawley, and P. M. Ferreira. Polynomial-complexity deadlock avoidance policies for sequential resource allocation systems. *IEEE Transactions on Automatic Control*, 42(10):1344–1357, October 1997.

[4] J. Park, S. A. Reveliotis, M. A. Lawley, and P. M. Ferreira. A correction to the RUN DAP for conjunctive RAS presented in 'Polynomial complexity deadlock avoidance policies for sequential resource allocation systems'. *IEEE Transactions on Automatic Control*, 46:672, 2001.

[5] M. Žarnay. *Systém na podporu rozhodovania pre riadenie dopravných procesov [Decision-support system for transportation systems control]*. PhD thesis, Faculty of Management Science and Informatics, University of Žilina, January 2007. In Slovak.

[6] M. Žarnay. Solving deadlock states in model of railway station operation using coloured Petri nets. In G. Tarnai and E. Schnieder, editors, *Formal Methods for Automation and Safety in Railway and Automotive Systems*, pages 205–213. L'Harmattan, October 2008.

[7] M. Žarnay. Banker's algorithm implementation in CPN Tools. In K. Jensen, editor, *Ninth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pages 123–142, October 2008.

[8] S. A. Reveliotis. Conflict resolution in AGV systems. *IIE Transactions*, 32(7):647–659, 2000.

[9] K. Jensen and L. Kristensen. *Coloured Petri nets. Modelling and validation of concurrent systems*. Springer-Verlag, July 2009.

[10] CPN Tools home page. Available at: <http://www.daimi.au.dk/CPNTools/>, online.

[11] J. L. Peterson. *Operating system concepts*. Addison-Wesley, 1981.

[12] F. Tricas. *Deadlock Analysis, Prevention and Avoidance in Sequential Resource Allocation Systems*. PhD thesis, Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, May 2003.

[13] T. Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, pages 541–580, April 1989.

[14] J. Park and S. A. Reveliotis. Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings. *IEEE Transactions on Automatic Control*, 46(10):1572–1583, 2001.

[15] E. G. Coffman, M. Elphick, and A. Shoshani. System deadlocks. *ACM Computing Surveys*, 3(2):67–78, June 1971.

[16] L. Jančík. Vyhýbanie sa uviaznutiam v systéme AGV [Deadlock avoidance in AGV system]. Master's thesis, Faculty of Management Science and Informatics, University of Žilina, May 2008. In Slovak.

[17] A. Kavička, V. Klima, and N. Adamko. Simulations of transportation logistic systems utilizing agent-based architecture. *International Journal of Simulation Modelling*, 6(1):13–24, 2007.

Acknowledgment

Research connected with this paper has been supported by the research projects MŠM 0021627505 – Theory of transport systems in the Czech Republic and VEGA 1/0361/10 Optimal design of public servicing systems in the uncertainty conditions.