Deadlock Avoidance in Railroad Operations Simulations

Jörn Pachl

Braunschweig : Institute of Railway Systems Engineering and Traffic Safety, 2011

Textfassung eines Vortrages auf dem 90th Annual Meeting des Transportation Research Board in Washington DC, 23-27 Jan 2011, Paper No. 11-0175

Veröffentlicht: 08.02.2011

http://www.digibib.tu-bs.de/?docid=00037775

DEADLOCK AVOIDANCE IN RAILROAD OPERATIONS SIMULATIONS
5681 words, 5 figures, 2 tables

AUTHOR

Prof. Dr. Joern Pachl
Technical University Braunschweig, Institute of Railway Systems Engineering and Traffic Safety
Pockelsstrasse 3, D-38106 Braunschweig, Germany
Phone: 49 531 391 3380, Fax: 49 531 391 5955, email: j.pachl@tu-bs.de

ABSTRACT
Powerful simulation software systems have become a standard tool for railroad capacity research. During a simulation run, deadlocks may occur at specific track configurations. A deadlock is a situation in which a number of trains cannot continue their path at all because every train is blocked by another one. This will make simulation results useless. For an adequate use of simulations, deadlocks should either be avoided completely or at least reduced to an acceptable level. Strategies against deadlocks developed in computer science cannot simply be adapted to railroad simulations. Also, typical algorithms developed in operations research for deadlock avoidance on railroads, do not really fit to the conditions of railroad operation. Now, a new solution has been developed that is based on a dynamic route reservation. That principle follows the idea that, before a train may be authorized to enter a track section, a specified number of track sections must be reserved ahead of that train. The number of track sections to be reserved varies depending on a set of logical rules. A track section can be reserved for several trains at the same time. These reservations are stacked in a similar way like stacked routes in an interlocking. The stacked reservations model the enforced train sequence that ensures a deadlock-free operation. That algorithm is currently being implemented into the Railsys simulation software and will be a standard feature in all new versions.

INTRODUCTION TO THE DEADLOCK PROBLEM

A deadlock is a self-blockade in a control system in which two or more tasks are waiting for each other to release a resource in a circular chain. In rail traffic, a deadlock is a situation in which a number of trains cannot continue their path at all because every train is blocked by another one (Figure 1). The ability to produce a deadlock depends on four necessary conditions:

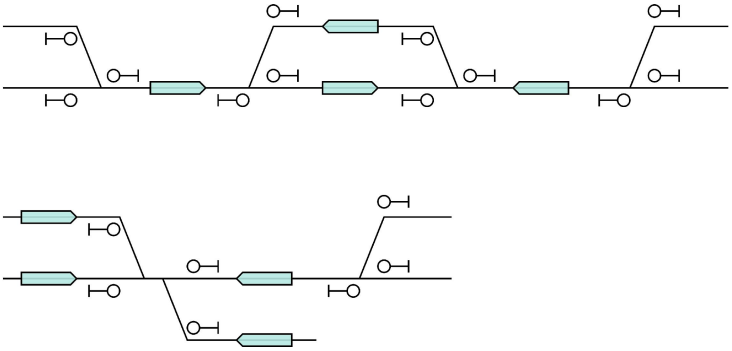

FIGURE 1 Simple deadlock examples

(1) The MUTUAL-EXCLUSION condition. That means that every element can be exclusively occupied by just one object at the same time. In a railroad system, the mutual-exclusion condition results from the fact that a section of track can exclusively be used by just one train at a time.

(2) The WAIT-FOR condition. That means that an object can wait for the release of an element. In a railroad system, that condition is obvious since trains may wait until a section is released.

(3) The NO-PREEMTATION condition. That means that an object can only occupy an element after that element has been released by the previous object. Or, in other words, it is not possible to remove objects from the system. That is obviously true for a railroad system.

(4) The CIRCULAR-WAIT condition. That means that the system must be able to produce closed circles of objects waiting for the release of elements currently occupied by other objects. The examples of Figure 1 show such circular waiting structures in a railroad system.

As a result, a railroad system has the ability to produce deadlocks. However, the actual deadlock probability depends significantly on the track layout. While the conditions (1), (2), and (3) are always in effect, the condition (4) depends on the structure of the track layout. In a layout without any track sections with bidirectional traffic, it is hardly possible to produce a closed chain of trains waiting for each other. Thus, on double track lines without bidirectional working, the deadlock probability is extremely low. On the other hand, on single track lines with a number of loop tracks for passing trains, operation may easily run into a deadlock.

In practical railroad operation, the timetable must never contain any deadlocks. As long, in a running operation, the scheduled train sequence is not changed, deadlocks will not occur. The deadlock problem becomes evident when, in case of delay, the scheduled train sequence is chanced by the dispatcher or when a railroad is operated on an unscheduled basis (typical on freight railroads in North America). In both cases, avoiding deadlocks is in charge of the dispatcher who controls train traffic. That is, why most automatic dispatching systems follow the timetable principle, i. e., in case of delay, they are altering the timetable in accordance with established scheduling rules.

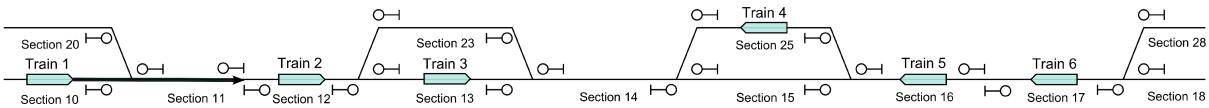

FIGURE 2 Infrastructure example of a congested single track line with two sidings

Avoiding deadlocks is a pure problem of event sequence. Once a chain of interdependencies has closed to a circle, there will be no way to escape the approaching deadlock. The deadlock problem is completely independent from speed and time factors. Figure 2 shows an example on a congested single track line with two sidings where trains of opposing directions can pass each other. In the shown situation, Train 1 is going to enter Section 11. Just by looking at this track chart, a positive decision can be made that the movement of Train 1 into Section 11 will necessarily lead into a deadlock. After Train 1 has entered Section 11, there are still several possibilities for the further development of the situation. So it is still an open decision which of the two trains waiting in Section 13 and Section 25 should enter Section 14 first. However, it is no longer possible to avoid a dead-

lock. That means that, in the current situation, Train 1 must not be allowed to proceed into Section 11. This statement can be positively made without knowing anything on speeds, running times, or priorities.

The development of strategies against deadlocks started in the late 1960s in computer science. The objective was to avoid system deadlocks when assigning resources to tasks in large computer systems. At the time, the basic classification of strategies against deadlocks was developed, which is still generally accepted and taught to students of computer science *(1) (2)*. There are three basic strategies against deadlocks:

- Deadlock detection and recovery
- Deadlock prevention
- Deadlock avoidance

With deadlock detection and recovery, deadlocks may occur but are automatically detected and removed either by terminating and restarting the involved tasks or by using a rollback mode in which the entire process is set back to a state before the deadlock occurred and then continued with an alternative event sequence. Both principles do not work in a railroad system. A task cannot be removed since this would mean to make a train disappear from the relevant track section. The rollback principle can be used in very simple transportation systems. In *(3)* it was proposed for automatic control of container cranes. In rail traffic control, setting back trains is not a practical solution. Due to the complexity of decision making in rail traffic, it is not even a useful method in simulations. In computer science, deadlock detection and recovery is still the leading strategy against deadlocks, however

Deadlock prevention means to remove at least one of the four necessary conditions for the deadlock ability. As an example, the wait-for condition will disappear if a task is only started after all necessary resources have been assigned to this task. In a railroad system this would mean that a train movement would only start after all signals have been cleared for this train through the entire territory. This is not the way a railroad works. Another example for deadlock prevention would be to have no track sections with bidirectional operation. Then, closed waiting loops can no longer occur and the circular-wait condition disappears. This is also not a way a railroad works.

Deadlock avoidance means to control a system in a way that a resource is never assigned to a task if that assignment will produce a closed waiting loop. In railroad operation, this means that a train must not be authorized to enter a track section if this movement leads into a deadlock. A typical example is the movement of Train 1 into Section 11 in Figure 2. In computer science, several algorithm for deadlock avoidance were developed. The best known are the Banker's algorithm and the algorithm of Habermann *(4)*. The basic idea of all these algorithms is that a task would block as many resources ahead as needed to safely avoid a deadlock. However, it may happen that more resources are blocked than really needed. This will increase the waiting time but reduce the processing time of the algorithm. Thus, in a computer system, there is more time saved than lost. In railroad operation, the situation is completely different. Holding trains unnecessarily back is not acceptable since it would significantly reduce capacity. As mentioned at the beginning, another method for deadlock avoidance is scheduling and solving all problems by rescheduling. As explained in the following chapter, the rescheduling doesn't work in synchronous simulations.

SIGNIFICANCE OF DEADLOCKS IN RAILROAD SIMULATIONS

Another field where deadlocks play an important role is railroad simulations used for capacity research. There are two basic types of simulation systems *(5), (6)*:

- Asynchronuos simulation
- Synchronous simulation.

Asynchronous simulation works quite similar to scheduling. It is possible to simulate the scheduling process separately from the process of a running operation. When simulating the scheduling process, the simulation system tries to put stochastically generated train paths into a timetable following the rules of scheduling. The different train classes are scheduled in order of their priority. For a given number of trains, the total of scheduled waiting time may be determined as the amount of time for which the train paths have to be postponed in order to get a timetable without scheduling conflicts (conflict = overlapping blocking times). The strategy of solving scheduling conflicts of two trains follows the rule of postponing the train path of the train with lower priority. When simulating the running operation, stochastic delays are generated which cause additional conflicts. These conflicts are solved with rules that follow the principles of real dispatching. In case of overlapping blocking times, train paths are postponed or the train sequence is changed in accordance to the priority of the trains. Thus, the simulation of running operation is done like a “disturbed scheduling”. Since asynchronous simulation always follows the scheduling principle, deadlocks will not occur.

In synchronous simulations, the situation is completely different since all partial processes of railroad operation are simulated in real time sequences. Typical examples of synchronous simulations are RTC in North America, and Railsys and Open Track in Europe. The results of synchronous simulations are quite close to the data to be expected in a real operation. With the help of a computer-based scheduling program, it is possible to

create timetables that could be evaluated by synchronous simulation. In contrast to asynchronous simulation, the process of scheduling cannot be simulated directly. Scheduling is only used to produce entry data of the simulation, but there is no parallel timetable processing during a running simulation. That is, why in case of delay, a synchronous simulation may produce deadlocks. Meanwhile, most simulation programs have a control logic that avoids very simple forms of deadlocks. However, on more complex infrastructures, especially track layouts with a lot of track sections with bidirectional operation, deadlocks are still a serious problem. Up to now, the only way to protect against deadlocks is to make sure that at the end of a simulation run, no deadlock that may have occurred remains undetected. This can be achieved when, after the end of the simulated period of time has been reached, the simulation is continued without generating new trains until all trains have left the territory. If the territory doesn't clear completely, some trains may have caught in a deadlock. If this happens just in a very few simulation runs, it will be still acceptable. There are known infrastructure examples, where almost every single simulation run will lead into a deadlock, however.

One possible solution of that problem was to integrate parallel timetable processing to the control logic. That principle required the simulation system to calculate a train path for every train from the current operational situation a sufficient time into the future. Or, in other words, it means to combine synchronous simulation with an asynchronous dispatching logic. The common statement of the makers of synchronous simulation software is that this principle is much too complicate and would consume too much processing power. While the last statement will probably disappear with coming computer generations, the first statement will still be true in the future.

Different from real rail traffic control in which deadlocks should always be safely avoided, there are the following requirements for deadlock avoidance in simulations:

• As long just a very few simulation runs fail due to deadlocks, it will not be a problem. Thus, the main objective of deadlock avoidance in simulations is to reduce the number of deadlocks to an acceptable level.

• The algorithm should be easy to implement. Algorithms of high complexity should be avoided.

• The preview ahead of a train should be limited to a distance absolutely necessary to avoid deadlocks.

• To ensure the highest possible degree of flexibility, the algorithm should request a final decision from the dispatching manager module on the use of sidings as late as possible.

• The algorithm should be able to handle situations to be found in real railroad operation, e. g., train routing in complex interlockings, and trains blocking more than one track section while waiting at a signal.

The Institute of Railway Systems Engineering and Traffic Safety at the TU Braunschweig has a long-term experience in applying the simulation software Railsys for capacity management. With increasing size of the simulated networks, the deadlock problem has become more and more evident. The experience from using Railsys has led to the following findings:

(1) Deadlocks do not occur everywhere where they are theoretically possible. Many networks have just a few deadlock sensitive spots while big parts of the network remain deadlock-free.

(2) The cycle length of almost all deadlocks is 2. Deadlocks with a cycle length of 3 exist only of the specific kind described under (3). Deadlocks with a cycle length of 4 are extremely rare and always related with very specific layouts. That supports the statements that the rule from computer science that more than 90 % of all system deadlocks have a cycle length of 2 is also expected to be true in rail networks.

(3) There is a specific kind of deadlocks that are caused by the fact that a train that is forced to stop at a signal doesn't fit with its entire length into the track section in approach to that signal so that the rear of the train doesn't clear the path of other trains (Figure 3). Such deadlocks are not as frequent as simple deadlocks with a cycle length of 2 but much more frequent than deadlocks with a cycle length of 4.

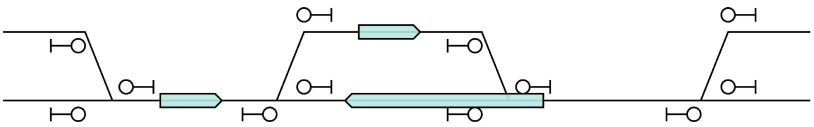

FIGURE 3 Deadlock caused by train length

The main reason that deadlocks with a cycle length of 4 are quite rare is that in a network with scheduled operation, traffic density on single track lines is kept on a level where, even in case of delay, the probability of four trains blocking a siding is very low. A situation as shown in the example of Figure 2, which was used to demonstrate the basic characteristics of a deadlock, will hardly ever occur in reality. Such a number of trains within such a close space would only be possible in an congestion caused by a temporare interruption of traffic. It will definitely not occur in a simulation that simulates traffic with typical delays. On railroads with unscheduled operation as typical on North American freight railroads, there is no scheduling process that prevents the traffic management from putting too many trains on a line. However, the typical train density on North Americ-

an freight railroads is far below the limits to have a chance of running into a very complex deadlock situation with four or more trains involved.

Despite the deadlock type described under (3), a cycle length of 3 can only occur in specific track layouts as triangles or loops. Such layouts do hardly exist in most real infrastructure examples. From the occurence of deadlocks that are caused by train length as described under (3), the lesson was learned that the deadlock avoiding algorithm must meet the condition that one train may occupy more then one track section at a time. Giving a train authority to enter a track section does not necessarily means that the train will clear the section it is currently occupying. Typical OR models do not meet that condition (in Chapter 3, that weakness is called the 'game board' philosophy).

KNOWN STRATEGIES AGAINST DEADLOCKS IN RAIL TRAFFIC CONTROL AND TRAFFIC SIMULATIONS

Different from computer experts, the interest of operations research experts in the deadlock problem started quite late. First papers on deadlocks in transportation systems were published in the 1980s. One of the first publications of that kind is the PhD thesis of Bruns *(3)* that doesn't concentrate on rail traffic but on general problems of automation in transportation systems. One of very first publications dealing with the deadlock problem in rail traffic control is the work of Petersen & Taylor *(7)*. The authors, who came from operations research, didn't use the term deadlock (they called it 'line block') and didn't cite any of the research on the deadlock problem that had been done before in computer science. The basic idea is an algorithm that evaluates the meetability of two opposing fleets of trains on a single track line. Later, other authors approved that the algorithm of Petersen & Taylor will safely avoid deadlocks. However, in specific situations, trains may be unnecessarily held back *(8)*.

In the early 1990s, based on the research being done in computer science and the early ideas for deadlock avoidance in rail traffic control, two different approaches were established. For traffic control of European railroads, deadlock avoidance by rescheduling became the leading solution. In complex systems, automatic rescheduling may become an extremely complex task, however. For the state of the art in rescheduling see *(9)*.

For synchronous simulations and also for traffic control on railroads with unscheduled operation, the rescheduling approach doesn't work. Here, algorithms for a rule-based deadlock avoidance have to be found. One of the first publications was *(10)* in which the authors provided illustrated examples of how, in case of delay, simulated traffic could run into a deadlock. They suggested to avoid deadlocks by adding logical decision rules. However, they did not develop a universal system of rules that could be applied to any infrastructure. Instead, they concentrated on the development of an interface that allows the user to add customized rules designed for individual locations. The reason for not developing universal rules for deadlock avoidance was obviously the complexity of the problem. In 2006, the authors of *(11)* brought it to the point: "It is very complicated to set up rules of dispatching to avoid deadlock situations, if the rules are not made very restrictive and cereby exclude certain events that may normally occur without causing any problems." For real-time dispatching they recommend the implementation of simplified rules. For interlocking simulators, solutions exist that can detect and solve conflicts, including deadlocks, within a single interlocking *(12)*. These solutions were not developed for dispatching and cannot avoid deadlocks on a longer stretch of line, however. Other authors tried to reduce the complexity of dispatching rules by Fuzzy algorithm *(13)* but never made it into a working system. Since the deadlock problem is purely deterministic, the Fuzzy approach won't work for that purpose.

In *(14)*, the author develops an approach for rule-based deadlock avoidance, which was later called Movement Consequence Analysis. That method was not developed for simulations but for automatic route setting systems (ARS) to be used on networks with stochastic, i. e., completely unscheduled operation. In Europe, that kind of operation is typical for a number of industrial railroads, e. g. for railroads in lignite coal opencast mining areas. Later, the same approach was also discussed to be integrated in the control logic of simulation systems. The basic idea is to analyze, which consequences will necessarily result from a train movement into a track for the further sequence of train movements. The consequences of a train movement are other train movements to be done. By this principle the consequences of a train movement can themselves produce further consequences and so on.

There are two different types of consequences. One type of consequences results from the fact that a train can only enter a section of track after both the last train ahead and the last opposing train which cannot be given way at a later point have cleared that section. Another type of consequences are train movements that must be made after a train has entered a section . This is always the case when a train enters the path of an opposing train that has now to wait at a point where the two trains can pass each other. In such a situation the first train must proceed until having reached a point where it clears the path of the blocked opposing train. If such a point does not exist, the movement of the first train will lead into a deadlock. After having entered the path of a train following in the same direction, the first train must be able to proceed at least one section. Thus, that kind of consequences force the first train to proceed.

Before a train is allowed to enter a track section, all consequences that will result from that train movement are analyzed. Deadlocks will lead to closed structures (loops) within the consequence chain. A loop occurs if, at any point of the chain, a train movement leads into the same section as the original train movement that is analyzed. If the original train movement was made, that train would be forced to leave that section. Doing this will produce the same consequences again and again. Figure 4 shows a simple example of the Movement Consequence Analysis for a train movement that would lead into a deadlock at a siding on a single track line. Since a train movement may have several consequences, in more complex situations, the consequence chain may develop into a tree structure. For examples see *(15)*. For use in synchronous simulation programs, Movement Consequence Analysis has several disadvantages. First is that this approach requires the detailed route of every train, i.e., the sequence of track sections the train is going to run through, to be already fixed. In a synchronous simulation, this is normally not the case since the places where trains will pass each other at sidings are flexibly determined while simulating the running operation. Also, it is still unclear, what preview length ahead of a train is needed to safely avoid deadlocks.

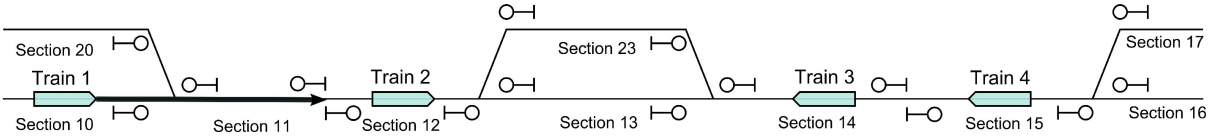

FIGURE 4 Example on a single track line

TABLE 1 Movement Consequence Analysis for the example of Figure 4

Movement	Consequence	Comment
Train 1 into **Section 11**	Train 1 into Section 12	Section 11 in path of opposing trains
Train 1 into Section 12	Train 1 into Section 13 Train 2 into Section 13	Section 12 in path of opposing trains Last train ahead has to clear section
Train 1 into Section 13	Train 2 into Section 14	Last train ahead has to clear section
Train 2 into Section 14	Train 4 into Section 23	Last opposing train has to clear section
Train 4 into Section 23	Train 3 into Section 12 Train 2 into Section 13	Last train ahead has to clear section Last opposing train has to clear section
Train 3 into Section 12	Train 3 into **Section 11** Deadlock!	Section 12 in path of opposing train

In 2003, as a result of a research project of the Australien Cooperative Research Centre for Railway Engineering and Technologies (Rail CRC), two papers were published that described deadlock avoiding algorithms for use in automatic dispatching on long single track lines. In *(16)*, the authors, both mathematicians, took the algorithm of Petersen & Taylor *(7)*, analyzed its weaknesses and developed a new, vector-based algorithm called the 'labelling algorithm'. For that, they also used new research results from deadlock avoidance in industrial engineering *(17)*. The developed model combines the original pure algorithmic approach with rescheduling. However, it remains at a very academic level. As *(18)* proves, even in quite simple situation, the proposed algorithm may lead to extremely complex graph model with a very complicate mathematical representation. In *(19)*, the original was developed into a more practical pure rescheduling-based model, which allows a flexible rescheduling. The authors of that paper also suggested using this approach for capacity research.

From all these models, one may get the impression that a lot of models already exist so that the only problem is to select the most appropriate approach for implementation into simulation software. However, all of these models are pure academic ideas. None of these models has ever been used in the control logic of real dispatching or simulation systems. The reason is that all of these models are based on the 'game board' philosophy, which is very typical for many models in operations research. In that philosophy, the railroad network is represented by a graph that reminds to a game board on which the trains move like play stones from field to field. Real railroad operation doesn't exactly meet that model. On a game board, each stone occupies exactly one position field. Unlike a play stone, a train can occupy several track sections at once. Entering a new track section doesn't

necessarily mean to clear the section the train is currently occupying. Thus, algorithms based on the game board model will not avoid deadlocks of the kind as shown in Figure 2. Also, in automatic block territory, the number of block signals may differ in both directions. This would result in a different number of position fields in both directions. This would not fit the game board model.

To overcome this and some other weaknesses of the known models, two new approaches to handle the deadlock problem in railroad operation emerged in the very last time. One is the idea described in the PhD thesis of Yong Cui *(8)*. Yong improved the Banker's algorithm to make it work under practical railroad conditions. Up to now, Yong's algorithm has only been demonstrated on very simple examples. There is still some research needed to approve usability in larger networks. The second approach is a model called Dynamic Route Reservation. A first sketch of that idea was described in *(15)*. At the time, it had not yet been tested and still contained some weaknesses, which were discovered later. In the meantime, the method has been improved and is now ready for implementation. The next chapter described the key idea of that method.

DEADLOCK AVOIDANCE BY DYNAMIC ROUTE RESERVATION

Dynamic route reservation follows the idea that, before a train may be authorized to enter a track section, a specified number of track sections must be reserved ahead of that train. The number of track sections to be reserved depends on specific rules described below. A track section can be reserved for several trains at the same time. These reservations are stacked in a similar way like stacked routes in an interlocking. The stacked reservations model the enforced train sequence that ensures a deadlock-free operation. To safely avoid a deadlock, a train must not enter a track section unless the train has reserved that section at the first position. Since the dispatching logic of most synchronous simulation programs is based on the reservation or on the pre-occupying of track sections ahead of the trains, the principle of modeling train sequences by reservation of track sections fits perfectly to the internal logic of such programs.

The reservation process follows these rules:

Rule 1: Before a train is authorized to enter a track section, the route into that section must be reserved.

Rule 2: If a route is reserved that leads into a track section with bidirectional operation where the train cannot give way to opposing trains, the route to leave that section must also be reserved.

Rule 3: If a route is reserved that leads into a track section that cannot take the entire train length, the route to leave that section must also be reserved.

Rule 4: If a track section to be reserved is still reserved for or occupied by another train, the reservation for that section is stacked to the next position.

Rule 5: If a reservation is put into the stack behind another train, for that train, if not yet the case, the route to leave that section must be reserved.

Rule 6: If a reservation is put into the stack behind another reservation that belongs to the current reservation process, i. e., that was initiated by the same original train, the stack positions of these two reservations must be exchanged.

TABLE 2 Dynamic Route Reservation for the example of Figure 4

Section	10	20	11	12	13	23	14	15	16	17
1st Position	Train 1>	<Train 3	<Train 3	Train 2>	Train 2>	<Train 3	<Train 3	<Train 4	Train 2>	
2st Position			Train 1>	<Train 3	Train 1>	<Train 4	<Train 4	Train 2>		
3rd Position				Train 1>			Train 2>			

Rule 6 is needed to avoid closed loops in the reservation processing, which may occur when a train, the reservation of which has been initiated by another train in the stack in accordance with Rule 5, is going to proceed into the same section the other train comes from. Table 2 shows a sheet with the reservations produced by these rules for the example of Figure 4. The current train positions are marked gray. In Section 11, the reservation for Train 1 is in the stack behind opposing train 3. Thus, if Train 1 moved into Section 11 without waiting for train 3, this would lead to a deadlock.

To overcome the weaknesses of models based on the game board philosophy, a new principle to model position fields was developed. When designing the logical infrastructure model, a position field is placed in every track section limited by two signals in at least one direction. If between two position fields a separating signal exists only in one direction, for the other direction, the two position fields are logically connected (Figure 5). That means, for that direction, the connected position fields do always have the same content. When a route is reserved for a train, the train is assigned to all position fields up to the next signal. After a train has completely passed a signal or, at railroads that use block overlaps, the clearing point behind a signal, that train is

cleared from the position field in approach to that signal. Logically connected position fields are always cleared together. When a position field has been cleared, trains in the stack move up to fill the cleared position.

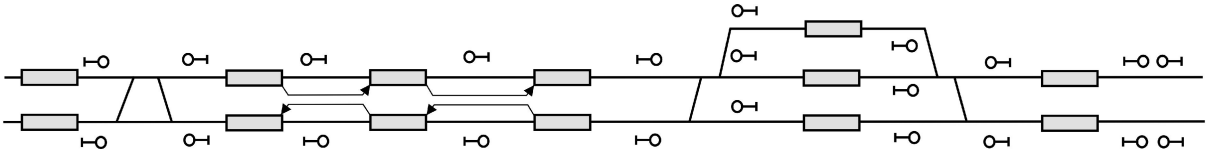

FIGURE 5 Connected position fields

The reservation process produces stacked routes that can be processed in the stacked order. When the dispatching manager of the simulation system requests a route for a train movement, a reservation process is initiated if the requested route has not yet been reserved as a result of Rule 2 or Rule 3 in earlier reservation processes. If the reservation results in having that route reserved at the first position, it can be used for the train movement. Otherwise, the requested route is rejected to be used for a train movement right now but remains in the stack. After the train has used the route and cleared the relevant sections, the route is removed from the first position. If stacked routes exist for that section, the next stacked route is moved to the first position and is now ready to be used for a train movement. Since this principle works very similar like the handling of stacked routes in a CTC system, the same procedure could also be used for automation of real CTC systems with priority-based automatic route setting.

CONCLUSIONS AND FURTHER PROSPECTS

With increasing size and complexity of territories simulated with railroad operations simulations, the deadlock problem has become more and more evident. Known strategies against deadlocks known for decades from computer science and operations research do not really fit to the specific needs of railroad operations simulations. With dynamic route reservation, a new approach exists that avoids the weaknesses of earlier methods but allows easy integration into the control logic of simulation programs. Up to now, the developed algorithm has been approved by manually applying it to various track arrangements of different characteristics and complexity. It is currently being implemented into the control logic of the simulation software Railsys. First experiences of using it in large territories with complex and high density operations are to be expected in the near future.

REFERENCE

1. Coffmann, E. G. jr.; Elphik, M. J.; Shoshani, A.: *System Deadlocks*. in: Computing Surveys 2(1971) S. 67-78

2. Bruns, M.: *Zum Entwurf von Steuerungen für komplexe Transportsysteme*. Diss. RWTH Aachen 1983

3. Havender, J. W.: *Avoiding deadlock in multitasking systems*. in: IBM SYST 2(1968) S.74-84

4. Habermann, A. N.: *Prevention of System Deadlocks*. in: Communication of the ACM 7(1969) S. 373-385

5. Pachl, J.: *Railway Operation and Control*. 2nd edition, VTD Rail Publishing, Mountlake Terrace, WA, 2009

6. Hansen, I. A.; Pachl, J. (eds.): *Railway Timetable & Traffic*. Eurailpress Hamburg 2008

7. Petersen, E. R.; Taylor, A. J.: *Line Block Prevention in Rail Line Dispatch and Simulation Models*. Information Systems and Operations Research, vol. 21 (1983), no. 1, S. 46-51.

8. Yong Cui: *Simulation-Based Hybrid-Model for a Partially-Automatic Dispatching of Railway Operation*. Dissertation. Institut für Eisenbahn- und Verkehrswesen der Universität Stuttgart, 2009

9. Jacobs, J.: *Rescheduling*. In: Hansen, I.; Pachl, J. (Hrsg.): Railway Timetable & Traffic. Eurailpress, Hamburg 2008

10. Tomii, N.; Satoh, N.: *A Train Traffic Simulation System Permitting Application of Knowledge Engineering*. Quarterly Report of RTRI, Vol. 31, No. 2, May 1990

11. Landex, A.; Kaas, A. H.; Hansen, S.: *Railway Operation. Centre for Traffic and Transport*. Technical University of Denmark, Kongens Lyngby 2006

12. Parádi, F.; Szilva, E.: „Konflikterkennung und –lösung durch dynamische Zuglenkung“. *Signal und Draht* 90(1998)3, p. 26-27

13. Fay, A.: *Wissensbasierte Entscheidungsunterstützung für die Disposition im Schienenverkehr*. Dissertation, TU Braunschweig, VDI-Verlag, Düsseldorf, 1999

14. Pachl, J.: *Steuerlogik für Zuglenkanlagen zum Einsatz unter stochastischen Betriebsbedingungen.* Dissertation. Schriftenreihe des Instituts für Verkehr, Eisenbahnwesen und Verkehrssicherung der TU Braunschweig, Vol. 49, Braunschweig 1993

15. Pachl, J. *Avoiding Deadlocks in Synchronous Railway Simulations.* 2nd International Seminar on Railway Operations Modelling and Analysis. March 28 - 30, 2007, Hannover, Germany. Proceedings CD-ROM

16. Mills, G.; Pudney, P.: *The Effects of Deadlock Avoidance on Rail Network Capacity and Performance*. Proceedings of the 2003 Mathematics-in-Industry Study Group

17. Valkenaers, P.; Van Brussel, H.: *Deadlock avoidance in flexible flow shops with loops*. Journal of Intelligent Manufacturing 14(2003), S. 137-144

18. Braaksma, A.: *Deadlock problems in Railway scheduling*. Traineeship at School of Mathematics and Statistics, University of South Australia, Adelaide 2008

19. Pudney, P.; Wardrop, A.: *Generating Train Plans with Problem Space Search*. University of South Australia, Adelaide 2006