Jörn Pachl

Avoiding Deadlocks in Synchronous Railway Simulations

Zuerst erschienen in:
2nd International Seminar on Railway Operations Modelling and Analysis.
28 - 30 March, 2007, Hannover, Germany. Proceedings CD-ROM

Avoiding Deadlocks in Synchronous Railway Simulations

Jörn Pachl
Technical University at Braunschweig, Institute of Railway Systems Engineering and Traffic Safety
Pockelsstrasse 3, 38106 Braunschweig, Germany, e-mail: j.pachl@tu-bs.de

Abstract
A deadlock is a situation in which a number of trains cannot continue along their path at all because every train is blocked by another one. In practical railway operation, deadlocks are avoided by the timetable and, in case of delay, by dispatching decisions of human traffic controllers. Avoiding deadlocks is also an essential feature of software systems for railway simulation. In simulation systems that follow the asynchronous principle, deadlocks will not occur since all conflicts are solved by rescheduling. In synchronous simulation systems, a positive strategy against deadlocks does not yet exist. The control logic of existing systems can only avoid simple forms of deadlocks. However, traffic simulation on critical infrastructures with a high probability of deadlocks is still a problem. The paper describes two principles of how deadlock avoiding rules could be integrated in the control logic of synchronous simulation systems and discusses the pros and cons of these two approaches.

Keywords
Railway simulation, deadlocks

1 Introduction to the Deadlock Problem

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). 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 ability to produce a deadlock depends on four necessary conditions:

(1) The MUTUAL-EXCLUSION condition. That means that every element can be exclusively occupied by just one object at the same time. In a railway 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 railway 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 railway systems.

(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 railway system.

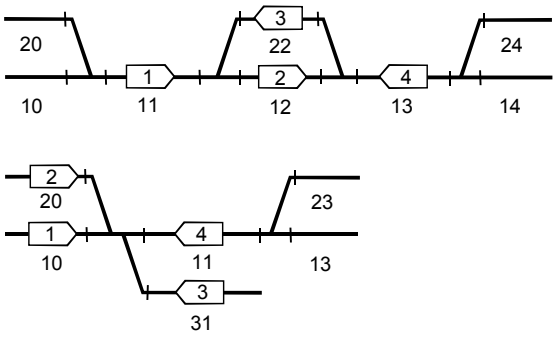

Figure 1: Simple deadlock examples

As a result, a railway 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 railway operation, the timetable must never contain any deadlocks. As long, in a running operation, the scheduled train sequence is not changed, deadlocks will not occure. The deadlock problem becomes evident when, in case of delay, the scheduled train sequence is chanced by the dispatcher or when a railway is operated on an unscheduled basis (typical on freight railways 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.

Another field where deadlocks play an important role is railway simulation used for capacity research. There are two basic types of simulation systems [1]:

- asynchronuos simulation,
- synchronous simulation.

Asynchronous simulation works quite similar to scheduling. It is possible to simulate the scheduling process apart 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. It is simply not possible to find a train path that will lead to a deadlock.

In synchronous simulation the situation is completely different since all partial processes of railway operation are simulated in real time sequences. Because of that, 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. There are known infrastructure examples, where even the most powerful simulation products fail completely, since every single simulation run will lead into a deadlock.

One possible solution of that problem was to integrate parallel timetable processing to the control logic. That principle required 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.

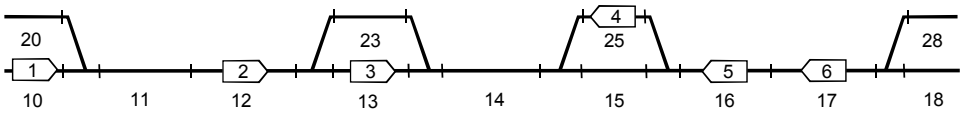

Figure 2: Example on a single track line

The question is, if there could be found a way to avoid deadlocks by synchronous dispatching rules that could easily be integrated into the control logic of a synchronous simulation program. At stated at the very beginning, avoiding deadlocks is a pure problem of event sequence. Due to that fact, the answer to that question is obviously yes. The dead-

lock problem is completely independent from speed and time factors. Figure 2 shows an example on a single track line with two intermediate stations 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 deadlock. That means that, in the current situation, Train 1 must not be allowed to proceed into Section 1. The following chapters describe two methods of how a definite decision can be made if the desired movement of a train into a track section will necessarily lead into a deadlock or not. These methods can be implemented into the dispatching logic of the simulation software.

For simulation software, the deadlock problem became evident in the early 1990s with the occurrence of powerful simulation systems. In 1990, Tomii and Satoh adressed the deadlock problem in railway simulations in [2]. They 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 customised rules designed for individual locations. The reason for not developing universal rules for deadlock avoiding was obviously the complexity of the problem. In 2006, the authors of [3] 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 [4]. These solutions were not developed for dispatching and cannot avoid deadlocks on a longer stretch of line. Other authors tried to reduce the complexity of dispatching rules by Fuzzy algorithm [5]. However, since the deadlock problem is purely deterministic, the Fuzzy approach won't work for that purpose.

Thus, the development of solutions for deadlock avoiding in synchronous simulation or real-time dispatching systems is still a big challenge. The following sections describe two solutions and discuss their pros and cons.

2 Movement Consequence Analysis (MCA)

The basic idea for the principle described here as Movement Consequence Analysis (MCA) was developed in the early 1990s for a completely different reason. At the time, it was part of a research project for the development of automatic route setting systems (ARS) that could be used on networks with stochastic, i.g. completely unscheduled, operation [6] [7]. In Europe, that kind of operation is typical for a number of industrial railways, e.g. for railways in lignite coal opencast mining areas. Now, it has been developed into a separate operations research model that can be used for different purposes, e.g. for evaluation of operating situations, and as a support tool for rescheduling algorithms. Deadlock avoiding in synchronous simulation is just one possible application.

The basic idea of MCA is to analyse which consequences will necessarily result from

a train movement for the further sequence of train movements. Therefore, a train movement is described by the movement of a train into a specific section of track. The consequences of a train movement are again train movements described in the same form. By this principle the consequences of a train movement can themselves produce further consequences and so on.

There are two different types of consequences. Train movements that must be made to enable a train to enter a section are called primary consequences. Train movements that must be made after a train has entered a section are called secondary consequences. Although, from the viewpoint of time order, the primary consequences happen before the concerned train movement can be made, they are logical consequences of that train movement. When a train, to enter a section, has to push forward other trains, the enforced movements of these trains are a consequence of the movement of the first train.

2.1 Primary Consequences

Primary consequences result directly from the fundamental rules of safe train separation, which are:

(1) A train can only enter a section of track after the last train ahead has cleared that section.

(2) A train can only enter a section of track after the last opposing train which cannot be given way at a later point, has cleared that section.

Figure 3 shows a track chart with typical examples. As an example for (1), the movement of Train 1 into Section 13 requires Train 2 to move into Section 12. An example for (2) is, that the movement of Train 3 into Section 12 requires Train 2 to move into Section 21.

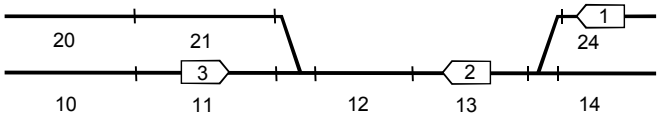

Figure 3: Example track chart

2.2 Secondary Consequences

The movement of a train into a track section may block the movement of other trains. 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 be able to reach 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. I.g., secondary consequences are always consequences for the concerned train itself. In the example of Figure 3, the movement of Train 3 into Section 12 requires not only Train 2 to move into Section 21 (primary consequence) but also Train 3 to move into Section 13 and then into Section 14 (secondary consequences). The reason is that, while beeing in Section 12 and 13, Train 3 blocks the path of opposing Train 1. The path of Train 1 can only be cleared by the movement of Train 3 into Section 14.

2.3 Consequence Trees

The example of the movement of Train 3 into Section 12 shows very clearly that one movement can have both primary and secondary consequences. Thus, analysing the consequences of a movement may lead to a tree structure. For the example of Figure 2.1, the MCA for the movement of Train 1 into Section 13 looks like this:

```
T1(S13) -> T2(S12)
T1(S13) -> T1(S12)
T1(S12) -> T2(S21)
T1(S21) -> T2(S20)
```

Deadlocks will lead to closed structures (loops) within that consequence tree. If the tree is used to analyse the consequences of a desired train movement, a loop occurs if, at any point of the tree, a train movement leads into the same section as the original train movement that is analysed. If the original train movement was made, that train would be forced to leave that section. Doing this will produce the same tree again and again.

Before starting with more complex examples, this principle is demonstrated at very simple deadlock examples similar to examples shown at the beginning. For the example of Figure 4, here is the analysis of the movement of Train 1 into Section 11:

```
T1(S11) -> T1(S12)
T1(S12) -> T2(S13)
T2(S13) -> T4(S22)
T4(S22) -> T3(S11) Deadlock
```

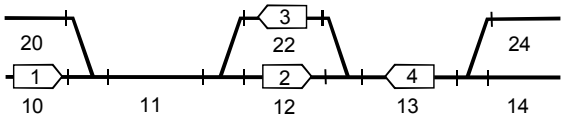

Figure 4: Example on single track line with one loop station

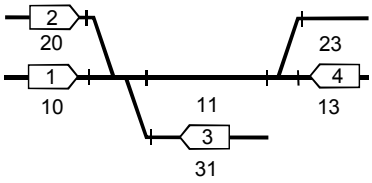

Figure 5: Example with crossing interlocking routes

In the example of Figure 5, the MCA depends on the destinations of the trains. If the destinations are:

Train 1 to Section 31,
Train 2 to Section 23,
Train 3 to Section 20,
Train 4 to Section 10,

then the MCA for the movement of Train 4 into Section 11 will look like this:

```
T4(S11) -> T4(S10)
T4(S10) -> T1(S31)
T1(S31) -> T3(S20)
T3(S20) -> T2(S11) Deadlock
```

With such still very simple examples, there is no real need for diagramming the tree structure. However, for more complex situations, tree diagrams are quite helpful. A typical example is the situation shown in Figure 6.

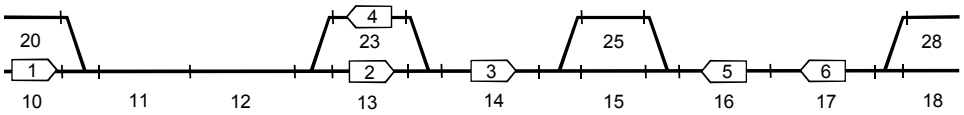

Figure 6: Example on single track line with two loop stations

The MCA for the movement of Train 1 into Section 11 looks like this:

```
T1(S11) -> T1(S12)
T1(S12) -> T1(S13)
T1(S13) -> T2(S14)
T2(S14) -> T3(S15)
T2(S14) -> T2(S15)
T2(S15) -> T3(S16)
T3(S16) -> T6(S25)
T6(S25) -> T5(S14)
T5(S14) -> T3(S15)
T5(S14) -> T5(S23)
T5(S23) -> T4(S12)
T4(S12) -> T2(S13)
T4(S12) -> T4(S11) Deadlock
```

One may wonder why the MCA analysis does not contain the following secondary consequences:

```
T3(S16) -> T3(S17)
T3(S17) -> T3(S18)
```

The reason is that Train 3 can occupy Section 16 only after Train 6 has reached Section 25. Thus, by running into Section 16, Train 3 is not entering the path of an opposing train. For a situation as complicate as the example of Figure 6, it is already very helpful to describe the MCA by a tree structure as shown in Figure 7.

To avoid misunderstandings, remember that this tree does not describe an event sequence but chains of dependencies. These logical chains do not represent the time order of events.

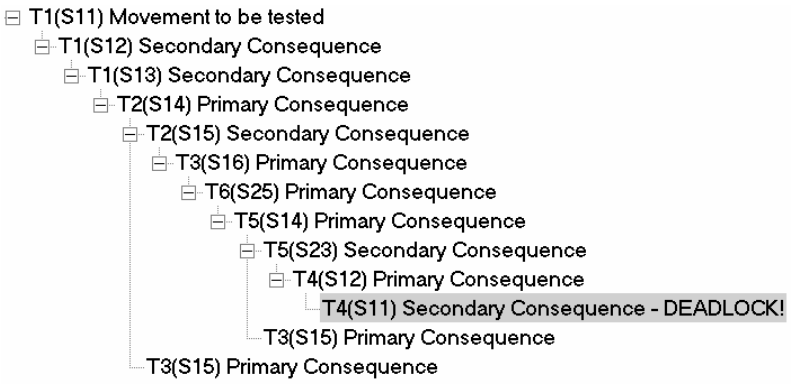

Figure 7: MCA tree structure for the example of Figure 6

Figure 8 shows a situation with the same trains on the same infrastructure but slightly different train locations. As a result, the MCA tree in shown in Figure 9 looks quite different from the MCA tree in Figure 7.

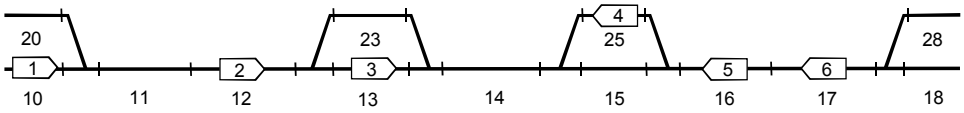

Figure 8: Modified situation from Figure 6

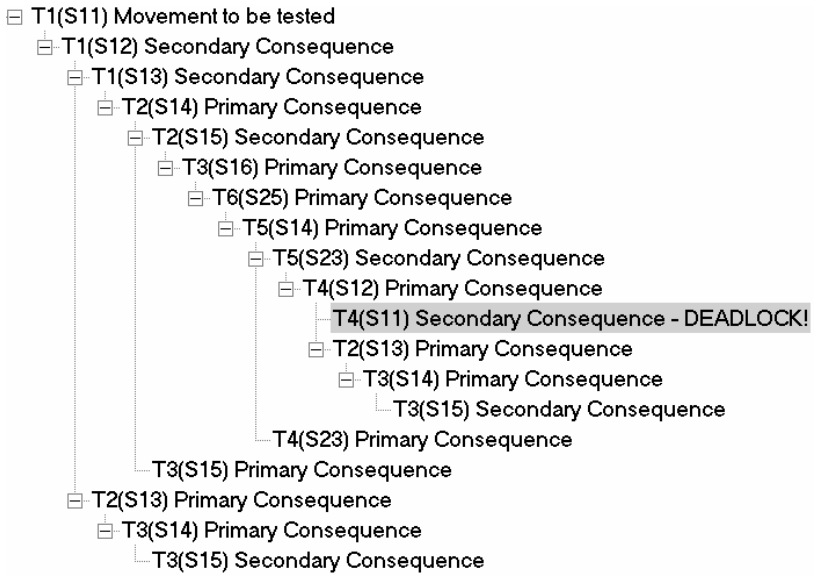

Figure 9: MCA tree structure for the example of Figure 8

For use in synchronous simulation programs, MCA has a number of disadvantages. First is that MCA requires that the detailed route of every train, i.g. the sequence of track sections the train is going to run through, is already fixed. In a synchronous simulation that is normally not the case since the places where trains will pass each other at loop tracks are flexibly determined while simulating the running operation. A possible solution is to use train routes that do not necessarily meet the real routes the trains will finally take. The routes used for MCA are just logical routes that use all tracks where opposing trains can give way to each other. This can be achieved when trains in passing loops always use the right track. In the running operation, at passing loops whithout a need for a meeting of opposing trains, the dispatching logic can flexibly change the use of tracks.

However, in stations with more than one loop track, that solution still lacks flexibility. Another problem is that it is not yet known how many track sections ahead of a train should be considered by the MCA. It makes obviously not much sense to detect a secondary consequence when a train is going to enter the path of an opposing train that is still very far away so that there is actually no chance for that train of being blocked by the desired train movement. Positive knowledge of the foresight required for MCA does not yet exist and may be subject of further research.

Another point is that the MCA principle does not fit very well into the control structure of some simulation systems. In these programs, implementation of MCA would make the system extremely complicate and would consume much processing time.

For these reasons, for integration in synchronous simulation program, another principle was developed that is described in the next chapter. However, it is based on some basic ideas of MCA.

3 Deadlock Avoiding by Dynamic Route Reservation (DRR)

The principle of deadlock avoiding by dynamic route reservation differs from MCA in two essential features. First, there are the following simplifications:

- When a train enters a section of track with bidirectional operation, except for a station track, the secondary consequence criterion is always in effect regardless of the actual locations of any opposing trains.
- When a train enters a station track section, the secondary consequence criterion is never in effect regardless of the actual locations of any opposing trains.

These simplifications allow a much greater flexibility in train routing. However, there are also two minor disadvantages. One is, that it is not absolutely guarantied that every deadlock is avoided. In difference to real railway operation, for a simulation program that point would be acceptable if just a very few numbers of simulation runs failed due to deadlocks. The second problem is that situations may occur, in which a train is unnecessarily held back (see the cite of [3] at the end of chapter 1). From what is known today, it is assumed that these situations are probably not really a problem since in most of such cases the concerned trains would be held back anyway for dispatching reasons. It is also possible to solve that problem by adding additional rules. Since these rules will make the entire procedure more complicate, implementation should only be considered if really needed.

The second difference to MCA is a completely different principle for modelling the train sequence dependencies. Dynamic route reservation follows the idea that before a train may be authorised to enter a track section, a specified number of track sections ahead of that train must be reserved. 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 preoccupying of track sections ahead of the trains, the principle of modelling train sequences by reservation of track sections fits much better to the internal logic of such programs than the MCA tree structures.

A train must not enter a track section

- if, according to the rules, that section cannot be reserved for that train, or
- if that section can be reserved for that train but not at the first position in the stack.

The reservation process follows these rules:

Rule 1: Before a train is authorised 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 other than a station track, the route to leave that section must also be reserved.

Rule 3: 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 4: 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 5: If a route to be reserved leads into a track that is reserved for or occupied by an opposing train, i.e. a train that will leave that section by moving into the section the route to be reserved comes from, the reservation of that route fails.

Rule 6: A reservation comes only into effect if all following reservations that are initiated by that reservation by rules 2 and 4 are possible. If the reservation of a route fails, all reservations previously initiated by rules 2 and 4 fail, too.

Now, these rules are demonstrated on a few examples. In the track charts, each track section has a number of reservation fields. The field within the track is the reservation with the highest priority. The adjacent fields are the following positions in the stack.

In SITUATION (1) of EXAMPLE A (Figure 10), reservations are made for Train 1 to leave Station C and for Train 2 to leave Station A. By Rules 1 and 2, these reservations lead into the station tracks of Station B. The two trains are allowed to proceed since their next sections are reserved at the first position.

In SITUATION (2) Train 1 and Train 2 have left their original stations. Reservation has been made by Rules 1 and 2 for Train 3 to leave Station C. By Rule 3, in two sections the reservation for Train 3 is in the stack position behind Train 1. As a consequence, Rule 4 forces to continue reservation for Train 1. By Rules 1 and 2, revervation for Train 3 has to continue into Station A. In two sections, the reservation for Train 3 gets into the second position behind Train 2. Since routes for Train 2 to leave these sections are already reserved, no further reservations for Train 2 are required.

In SITUATION (3) Train 3 has left Station C. Reservation has been made for a Train 4 to leave Station A. Following Rules 1 and 2, that reservation runs into the station track of

Station B. In all sections reserved for Train 4, the reservation is in the second or even the third position. That means, Train 4 is currently not allowed to leave Station A. Only after the arrival of Train 1 at Station A, the reservation for Train 4 in the first section right of Station A will move from the second to the first position. Then, Train 4 may proceed. The reservation made for Train 4 in the station track of Station B in the next stack position behind Train 2 forces to continue reservation for Train 2 into the station track of Station C. This prevents trains to enter the line from Station C.

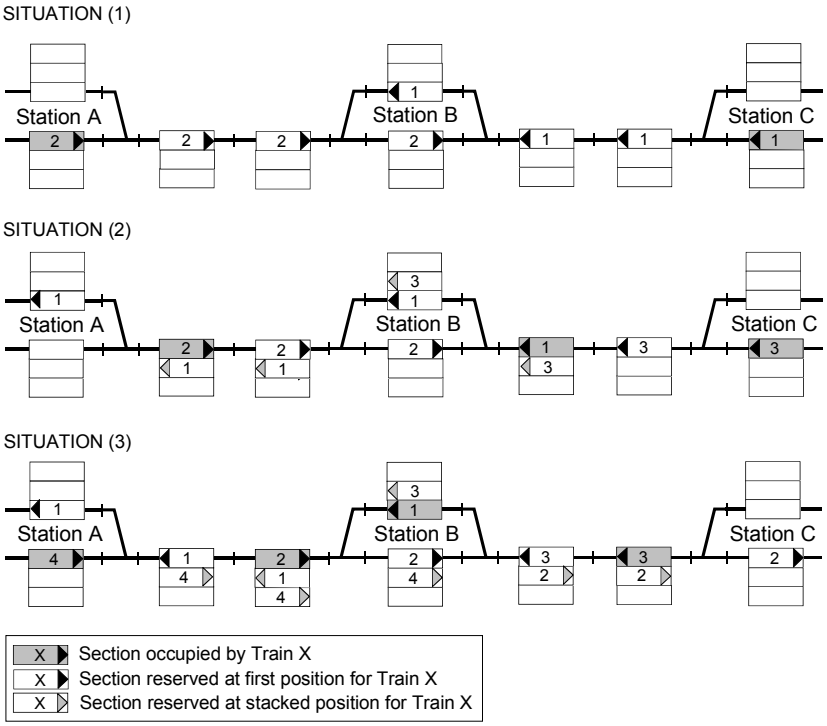

Figure 10: EXAMPLE A

In SITUATION (1) of EXAMPLE B (Figure 11), Train 1 is moving from Station C to Station B. Reservation is made for Train 3 to leave Station C and to follow Train 1 on the same route. By Rule 3, reservation for Train 1 must now continue into Station A. In SITUATION (2), reservation for Train 2 between Station A and Station B goes into the stack behind Train 1. In this case Train 2 is unnecessarily held back because the movement of Train 2 to leave Station A before Train 1 has left Station B will not lead into a deadlock. It is assumed that it would not really hurt to held back Train 2 in Station A since in the given situation the dispatcher would probably give Train 1 priority over Train 2 to minimise delay.

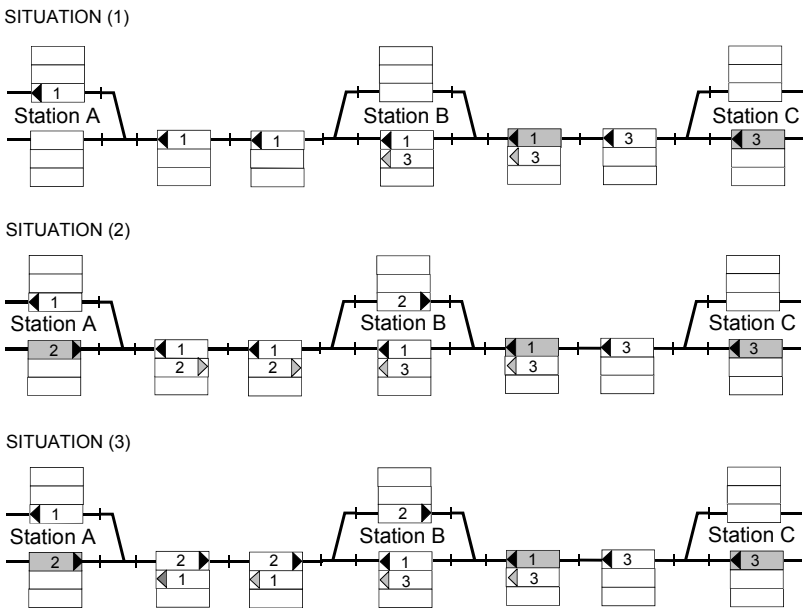

Figure 11: EXAMPLE B

If further research comes to the result that this assumption does not work, an additional rule could be added that allows changing the stack order in specified cases. Here is a draft for that rule:

Rule 7: If the reservation of successive routes for a train ends in a section where this train is in the first position of the stack, then in all intermediate sections reserved for that train, the reservation can be moved to the first stack position.

Applying Rule 7 to SITUATION (2) will change that situation into SITUATION (3). However, it is not yet sure if Rule 7 is really needed.

Figure 12 shows an example in which the movement of Train 2 to leave Station A is denied because the reservation required by Rule 2 is not possible due to Rule 5. Figure 13 shows another example with one more station involved. There, the reservation for Train 4 to leave Station A would lead into the track that is currently occupied by Train 2. By this, reservation for Train 4 requires to continue reservation for Train 2. Since this is not possible for the same reason as in Figure 11, reservation for Train 4 fails, too.

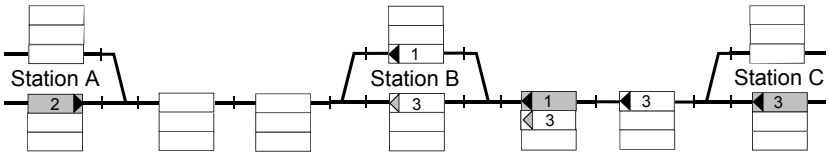

Figure 12: EXAMPLE C

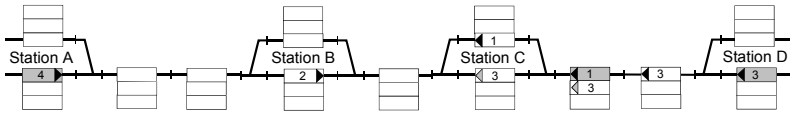

Figure 13: EXAMPLE D

EXAMPLE E (Figure 14) demonstrates how the reservation works on a line with a station with more than two station tracks. It also shows that the decision which station track should be assigned to a train must be made at least before the train leaves the last station in rear.

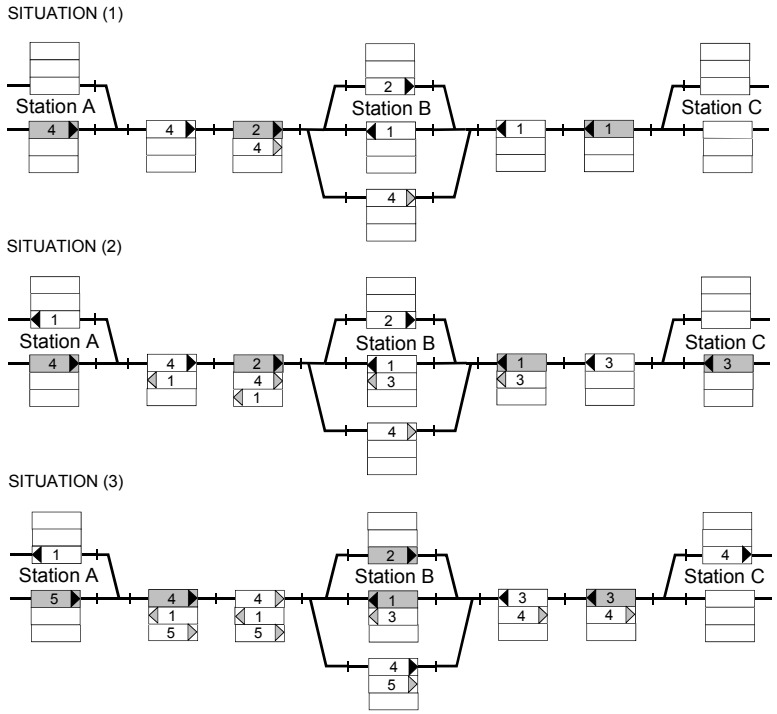

Figure 14: EXAMPLE E

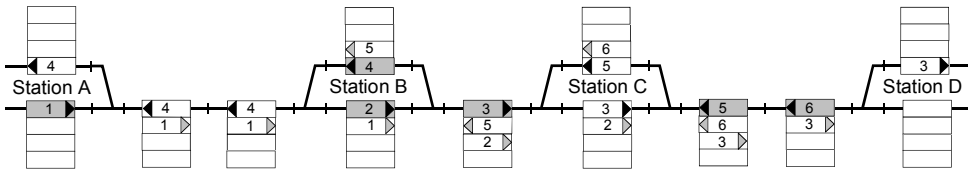

Figure 15: EXAMPLE F

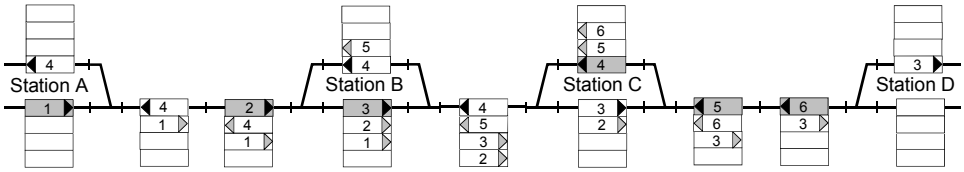

Figure 16: EXAMPLE G

Figures 15 and 16 show examples that meet the situations in Figures 6 and 8 which were used to explain MCA trees. In both examples reservation has just been made for Train 1 to leave Station A. In EXAMPLE G, Train 4 was given priority over Train 3 between Station B and C. This example also shows that for that line a stack with four positions is needed.

4 Conclusions

Avoiding deadlocks is an essential feature of railway simulation systems. In existing simulation systems that follow the synchronous principle, only simple forms of deadlocks are positively avoided. The paper describes two principles of how deadlocks of any complexity can be avoided in synchronous simulation. The two principles have their specific pros and cons. The first principle is called Movement Consequence Analysis (MCA). The basic idea of MCA is to analyse which consequences will necessarily result from a train movement for the further sequence of train movements. The dependencies between the consequences are modelled in logical tree structures. MCA can handle very complex situations but it leaks flexibility since the usage of station tracks has to be fixed in advance over the entire train path. The second principle is called Dynamic Route Reservation (DRR). DRR follows the idea that, before a train may be authorised to enter a track section, a specified number of track sections ahead of that train must be reserved. The number of track sections to be reserved depends on specific 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 route reservations describe the train sequence for a deadlock-free operation. This allows more flexibility than the MCA approach. However, from what is known today, it seems possible that in very rare situations, a deadlock may occur. For a simulation program that point would be acceptable if just a very few numbers of simulation runs failed. Therefore, the next steps in research will be to implement the described rules into the control logic and to test them on infrastructure examples of different operational characteristics.

References

[1] Pachl, J.: *Railway Operation and Control*. VTD Rail Publishing, Mountlake Terrace, WA, 2002.

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

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

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

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

[6] 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

[7] Pachl, J.: „Zum Deadlock-Problem bei der automatischen Betriebssteuerung“. *Signal und Draht* 89(1997)1/2, p. 22-25