Safety criteria for the vital processor interlocking at Hoorn–Kersenboogerd

Wan Fokkink
Department of Philosophy, Utrecht University,
Utrecht, The Netherlands

Abstract

We formulate several classes of safety criteria for railway yards in terms of observable behaviour. These criteria are meant to protect trains from collisions and from derailments. We identify a number of safety criteria, and present instances of these classes for the case of the railway yard at station Hoorn–Kersenboogerd. These criteria have all been checked by means of the Stålmarck theorem prover, using a methodology from Groote, Koorn and Van Vlijmen.

1 Introduction

At a sizable number of Dutch railway stations, among which station Hoorn–Kersenboogerd, computer equipment based on a Vital Processor Interlocking (VPI) is used in order to ensure safe movement of trains. Apart from a number of hardware checks, a VPI essentially executes a program that consists of a large number of assignments of the form $v = \phi$ with v a variable and ϕ a Boolean formula, which expresses dependencies between objects such as points, signals and level crossings, taking into account detailed information such as delays of electrical devices. Each railway station is supplied with its own set of assignments, based on the particularities of the road map of the railway tracks. We focus on properties of such sets of assignments.

Security in a railway yard is ensured using its Boolean formula as follows. Each second, the VPI executes a control cycle, which starts with reading new values for the input variables, which are determined by the environment. Next, these values are latched by the VPI, and they are used to compute the new values for the internal variables and for the output variables. Finally, the output values are transmitted to the outside world, where they are used in managing the signals, points and level crossings. After some idle time, in order to fill up the second, the VPI executes the next control cycle.

A specification of VPI, together with the verification of several desirable properties of VPI, is presented in [4]

The production of a set of assignments for a specific railway yard is an involved human business, and even for a small railway station, the resulting set of assignments is large and complicated. So a bottleneck in the strive for a hundred percent security of train movements guided by VPI, is the possible existence of flaws in the assignments. Hence, it is worthwhile to try and find a mechanized way to examine the assignments on the presence of defects. In order to build a sound testing environment for this purpose, it is desirable to have a classification of safety requirements for railway yards, together with a description how to obtain the requirements for each class from a particular layout of railway tracks. This paper constitutes a first attempt to formulate such a classification of safety criteria, in the case of the comparatively simple road map at railway station Hoorn–Kersenboogerd.

In cooperation with engineers from Holland Railconsult, several classes of safety requirements were detected. These safety criteria are all meant to protect trains from collisions and from derailments. For each class, we give examples of requirements that are generated in the case of the road map at Hoorn–Kersenboogerd. Each requirement is supplied with ample informal comments, so that it should not be necessary to have a thorough knowledge of the notations and concepts that are used in the assignments, in order to grasp the safety requirements. For a description of the full range of requirements for Hoorn–Kersenboogerd the reader is referred to [2].

Our classification does not cover the full range of desirable safety criteria, that is, validity of these criteria does not ensure that trains will never collide nor derail. However, the safety criteria that are described in this paper do make a satisfactory collection for a first testing system to check the safety of a set of assignments. Experience learns that often flaws in the assignments upset the validity of one of the safety requirements that are formulated in this paper. In particular, by automatic checking of our safety criteria, we have spotted mistakes in draft sets of assignments for railway stations Hoorn–Kersenboogerd and Heerhugowaard.

Assume a set of assignments S, related to some railway yard. By taking the conjunction of all the assignments, S can be turned into a large Boolean formula Φ. Certain requirements involve time, such as 'if a section has been unoccupied for ten seconds, then ...'. Hence, we want to describe the dependencies between the objects in the railway yard in a period of time. Therefore we make copies $\Phi_0, \Phi_1, ..., \Phi_n$ of Φ, where Φ_i describes the dependencies at the railway yard i seconds ago. Finally, the conjunction $\Phi_0 \wedge \cdots \wedge \Phi_n$ is the desired Boolean formula, which describes the dependencies between the objects in the railway yard in the last n seconds.

For each safety requirement, we want to be sure that it holds in all circumstances. Thus, in the case of a particular requirement R, we desire that the Boolean expression $(\Phi_0 \wedge \cdots \wedge \Phi_n) \Rightarrow R$ holds for all possible values

of variables in this expression. Experience learns that $\Phi_0 \wedge \cdots \wedge \Phi_n$ is in general too large to allow manipulations of such a Boolean expression on a computer with 'reasonable' capacity. Hence, as a first step, subformulas of $\Phi_0 \wedge \cdots \wedge \Phi_n$ that do not contribute to the value of R are removed by means of a slice algorithm [8], producing a, usually considerably smaller, formula Ψ. Finally, satisfiability of $\neg(\Psi \Rightarrow R)$ is checked using a theorem prover.

All the requirements that are presented in this paper, for the set of assignments for station Hoorn–Kersenboogerd, have been checked by means of the Stålmarck theorem prover. This tool can handle large Boolean formulas, by the application of smart algorithms for computations in classical logic. For information on innovative constructions that have been implemented in this theorem prover, see [5, 6]. In order to apply the Stålmarck theorem prover, we have used the Prolog interface NP Module [1]. We have also tried to check the safety requirements in an improved BDD based theorem prover [3], but requirements which involve time, such as 'if a signal shows red for one second', turned out to be too hard to handle for this tool.

The Stålmarck theorem prover has been applied before to verify interlocking equations in a computer controlled interlocking system used by the Swedish state railways [7].

2 Notations and basic concepts

2.1 The road map at Hoorn–Kersenboogerd

Basically, a railway yard consists of a collection of linked railway tracks, supplied with features such as signals, points and level crossings. Figure 1 depicts a schematic view of the road map of the railway yard at Hoorn–Kersenboogerd. In this figure, the objects 52D, 62A-C, 66A-C, 69A-B, 70A-C, 73A-B, 74A-B denote tracks, the objects 60, 62, 64, 66, 68, 70, 72, 74 denote signals, the objects 69, 73 denote points, and object 35.0 denotes a level crossing.

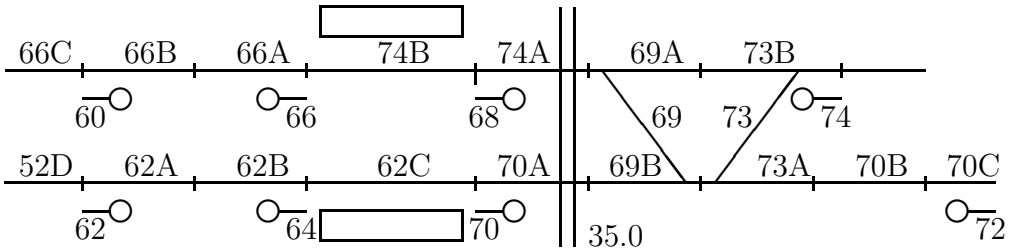

Figure 1: The road map at Hoorn–Kersenboogerd

2.2 Vital train logic

Each of the objects in a railway yard can attain a certain number of states:

- a railway track is either occupied or unoccupied,

- a signal shows either red or flashing yellow or yellow or green, possibly together with a number in order to impose a speed limit.
- points are either in reverse or in normal position, or in neither of both,
- a level crossing is either open or closed, or neither of both.

In order to avoid dangerous situations on the railway yard, certain combinations of states are to be avoided. For example, if a level crossing is open, then the track where it is situated must be unoccupied, in order to avoid a collision of a train with passing street traffic. In ViTaL (VTL), such dependencies are expressed by means of Boolean expressions. First, each possible state of an object in the railway yard is represented in the form of a Boolean variable, which is true if and only if this state is attained. For example, the state 'track 70A is unoccupied' is represented by the variable 70A_TR_DI, and the state 'level crossing 35.0 is open' is represented by the variable 350_XR_DBO. Then dependencies between states can be expressed by means of Boolean expressions. For example, 'if level crossing 35.0 is open, then track 70A is unoccupied' becomes $350\_XR\_DBO \Rightarrow 70A\_TR\_DI$.

VTL incorporates three kinds of variables:

- input variables, whose values are determined by the environment,
- output variables, which determine the states at the signals, points and level crossings in the railway yard,
- internal variables, which together with the input variables are used to determine the values of the output variables.

A set of assignments is constructed from variables, the conjunction denoted by &, the disjunction denoted by #, the negation denoted by ˜, and the implication denoted by $\Rightarrow$. (These notations are based on the notational conventions that are used in the Stålmarck theorem prover.) As a binding convention, negation binds stronger than conjunction and disjunction, which in turn bind stronger than implication.

2.3 Safety criteria

In order to build a sound testing environment for VTL, it is desirable to have a classification of safety requirements for railway yards. In this paper, we formulate several classes of safety criteria:

1. Dependency relations between signals; the aspect of a signal may yield restrictions on the range of aspects of a previous signal.
2. If a signal does not show red, then there is a *route* connected to this signal. That is, if a train passes this signal, the points behind this signal are positioned such that the train will reach the next signal without the possibility of being derailed.

3. If a signal does not show red for one second, then the counter-signals in its route are to show red.

4. If a signal does not show red nor flashing yellow for one second, and if the route connected with this signal overlaps with the route connected with some other signal, then that other signal must show red.

5. If a signal does not show red nor flashing yellow, then there is no train present on its route.

6. Points can only be reversed if they are unoccupied.

7. If points are part of a route, then their position is either normal or reverse.

8. If a level crossing is open, then tracks near the level crossing must be either unoccupied, or separated from the level crossing by a red signal.

Given a railway yard, we can produce a specific set of safety requirements by taking instantiations of the classes above. As an example, we shall describe in detail which safety requirements are generated in the case of the road map at Hoorn–Kersenboogerd.

2.4 Description of variables

In order to obtain classes of safety requirements that are independent of the specific VPI, we will make sure to use input and output variables only in the safety requirements. In the safety requirements, we shall encounter the following kinds of input variables, which all end on DI (Direct Input).

- TR_DI and TRPR_DI.

  Variables of this form denote that a track is unoccupied. For example, 62C_TR_DI is true if there is no train present on track 62C, and 66A_TRPR_DI is true if there is no train present on track 66A. (The distinction in notation expresses a distinction in the number of available contacts at the relay which detects whether the track is occupied. This distinction is of no importance in VTL.)

- RWP_DI and NWP_DI

  Variables of this form express whether points are in reverse or in normal position respectively. For example, 69_RWP_DI means that the points 69 are in reverse position.

In the safety requirements, we shall encounter the following kinds of output variables, which end either on ACO (Alternating Current Output) or on DBO (Double Break Output).

- R_ACO and GLFL_ACO and GL_ACO and GR_ACO.

  Variables of this form denote the aspect at a specific signal. For example, 74_R_ACO or 74_GLFL_ACO or 74_GL_ACO or 74_GR_ACO is true if signal 74 shows red or flashing yellow or yellow or green respectively.

- XR_DBO.

  Variables of this form express that a certain level crossing is open.

- RWR_DBO and NWR_DBO.

  Variables of this form indicate that points are being put into reverse or normal position respectively. (In theory, points can be instructed to go into reverse and normal position at the same time. In practice, this will never be the case; it would cause the points to break down.)

In the safety requirements, it is sometimes necessary to look back at the past. For example, we may want to know whether a signal showed red one second ago. Such values are expressed by variables with the suffix J_n, yielding the value of a variable n seconds ago. So the variable 74_R_ACO_J_1 is true if and only if the signal 74 showed red one second ago.

3 Dependency relations for signals

The aspect of a signal may cause restrictions on the range of aspects at previous signals. For example, if a signal shows red or flashing yellow, then in general the most liberal aspect for a previous signal is yellow. In the case of Hoorn–Kersenboogerd, an example of such a requirement is: 'if signal 70 shows red or flashing yellow, then signal 62 must show red or flashing yellow or yellow'. This can be expressed in terms of Boolean notation by

```
70_R_ACO # 70_GLFL_ACO => 62_R_ACO # 62_GLFL_ACO # 62_GL_ACO
```

The dependencies between signals are described on the 'OBE page'. The requirements that are imposed by dependencies between signals can be copied from this OBE page without any complications.

4 Routes

A *route*, connected to a signal, is the route that a train follows after it has passed the signal, until it reaches a next signal in the same direction. A route is determined by the position of the points on the track. There are three categories of requirements connected with routes:

1. If a signal does not show red, then there is a *route* connected to this signal. That is, if a train passes this signal, the points behind this signal are positioned such that the train will reach the next signal without the possibility of being derailed.

2. If a signal does not show red for one second, then the counter-signals in its route are to show red.

3. If a signal does not show red nor flashing yellow for one second, and if the route connected with this signal overlaps with the route connected with some other signal, then that other signal must show red.

These three categories of requirements for routes and signals are illustrated in the case of signal 74.

If signal 74 does not show red, then there must be some route from signal 74 to a following signal. That is, the points 73 and 69 are to determine a route from signal 74 onwards. There are three possible routes:

1. the points 73 are reverse and the points 69 are reverse,

2. the points 73 are reverse and the points 69 are normal,

3. the points 73 are normal and the points 69 are normal.

Note that the situation 'the points 73 normal and the points 69 reverse' is not possible, because then a train passing signal 74 would ride open the points 69. Thus, we obtain the following requirement.

```
~74_R_ACO => (73_RWP_DI & (69_RWP_DI # 69_NWP_DI))
             # (73_NWP_DI & 69_NWP_DI)
```

If signal 74 does not show red for one second, then the counter-signals in its route are to show red. Hence, we find the following requirements.

- Each possible route from signal 68 would cross each possible route from signal 74, so if signal 74 does not show red for one second, then signal 68 must show red.

  ```
  ~(74_R_ACO_J_1 # 74_R_ACO) => 68_R_ACO
  ```

- If the points 73 are reverse, then the routes from signal 74 and from signal 70 cross. So if signal 74 does not show red for one second, and if the points 73 are reverse, then signal 70 must show red.

  ```
  ~(74_R_ACO_J_1 # 74_R_ACO) & 73_RWP_DI => 70_R_ACO
  ```

If signal 74 does not show red nor flashing yellow for one second, then routes which overlap with the route of signal 74 must all be guarded by a red signal. This leads to the following requirements:

- Suppose that signal 74 does not show red nor flashing yellow for one second. Furthermore, let the route from signal 74 lead to signal 66, that is, let either the points 73 be normal, or let the points 73 and 69 be reverse. Then signal 60 must show red.

  ```
  ~(74_R_ACO_J_1 # 74_R_ACO # 74_GLFL_ACO_J_1 # 74_GLFL_ACO)
   & (73_NWP_DI # (73_RWP_DI & 69_RWP_DI)) => 60_R_ACO
  ```

- Suppose that signal 74 does not show red nor flashing yellow for one second. Furthermore, let the route from signal 74 lead to signal 64, that is, let the points 73 be reverse and the points 69 be normal. Then signal 62 must show red.

  ```
  ~(74_R_ACO_J_1 # 74_R_ACO # 74_GLFL_ACO_J_1 # 74_GLFL_ACO)
   & 73_RWP_DI & 69_NWP_DI => 62_R_ACO
  ```

In a similar fashion, safety requirements for routes can be formulated for the remaining signals. Some of these requirements, namely those imposed by the third category on signals 60 and 62, turned out to be invalid for the draft set of assignments for railway station Hoorn–Kersenboogerd which was placed at the disposal of Utrecht University.

5 Occupation of tracks

In this section we implement the requirement that if a signal does not show red nor flashing yellow, then there are no trains present on its route. Again, we only present the requirements in the case of signal 74. Assume that signal 74 does not show red nor flashing yellow.

Track 73B is part of each route from signal 74, so it must be unoccupied.

```
~(74_R_ACO # 74_GLFL_ACO) => 73B_TR_DI
```

If the points 73 and 69 are reverse, or if the points 73 are normal, then the route from signal 74 includes the tracks 69A and 74A and 74B. So these tracks have to be unoccupied.

```
~(74_R_ACO # 74_GLFL_ACO) & ((73_RWP_DI & 69_RWP_DI)
 # 73_NWP_DI) => 69A_TR_DI & 74A_TR_DI & 74B_TR_DI
```

If the points 73 are reverse, then the route from signal 74 includes the tracks 73A and 69B. So these tracks have to be unoccupied.

```
~(74_R_ACO # 74_GLFL_ACO) & 73_RWP_DI => 73A_TR_DI & 69B_TR_DI
```

If the points 73 are reverse and the points 69 are normal, then the route from signal 74 includes the tracks 70A and 62C. So these tracks have to be unoccupied.

```
~(74_R_ACO # 74_GLFL_ACO) & 73_RWP_DI & 69_NWP_DI
 => 70A_TR_DI & 62C_TR_DI
```

6 Points

While points are being put in reverse or in normal position, they have to be unoccupied. For the points 69 this leads to the following requirement.

```
69_RWR_DBO # 69_NWR_DBO => 69A_TR_DI & 69B_TR_DI
```

If points are part of a route, then they are either in normal or in reverse position. In the case of signal 74 and the points 69 this leads to the following requirements.

```
~74_R_ACO & 69_NWP_DI => ~69_RWP_DI
~74_R_ACO & 69_RWP_DI => ~69_NWP_DI
```

7 Crossings

If level crossing 35.0 is open, then track 74A has to be unoccupied, and either signal 68 shows red, or track 74B is unoccupied.

```
350_XR_DBO => 74A_TR_DI & (68_R_ACO # 74B_TR_DI)
```

Similarly, if level crossing 35.0 is open, then track 70A has to be unoccupied, and either signal 70 shows red, or track 62C is unoccupied.

```
350_XR_DBO => 70A_TR_DI & (70_R_ACO # 62C_TR_DI)
```

8 Conclusions

We have presented several categories of safety criteria, in VTL, and we have worked out what specific criteria are generated from these categories in the comparatively simple case of Hoorn–Kersenboogerd railway station. These criteria have been verified using the Stålmarck theorem prover.

Thus, we have obtained the routine to produce safety criteria for railway stations in VTL. As a next step, this routine is to be applied to other railway stations, with a considerably larger set of assignments than the one for Hoorn–Kersenboogerd railway station. For example, at the moment we are occupied with producing the safety criteria for the much more complicated situation at Heerhugowaard railway station.

Ideally, the routine to produce safety criteria should result in an algorithm which can be implemented, so that the safety criteria for a railway station can be produced automatically. Namely, in the case of Hoorn–Kersenboogerd there are few railway tracks, so that the routine yields few requirements, which can be produced by hand. But the VPI has also been installed at stations where there are many more distinct routes, which leads to an inevitable explosion in the number of safety criteria.

An important question is whether the categories of safety criteria that have been developed in this paper are sufficient to ensure absolute safety on railway tracks, meaning that trains will never collide nor derail. In fact, we can already give a negative answer to this question, because we are aware of dynamic safety criteria, which involve occupied routes, that we could not express properly in terms of VTL.

Acknowledgements

Gea Kolk, Peter Musters and Robert Straatman from Holland Railconsult provided valuable information on the technical subtleties of the assignments for Hoorn–Kersenboogerd. Jan Friso Groote and Bas van Vlijmen guided me into the world of VPI. Paul Berghege and Kiavash Shams helped to verify the safety requirements. Joost Mertens supplied useful comments.

References

[1] Andersson, S. NP Module, User manual, 1994.

[2] Fokkink, W.J. Safety criteria for Hoorn–Kersenboogerd railway station, Logic Group Preprint Series 135, Utrecht University, 1995. Available by ftp from `ftp.phil.ruu.nl` as `logic/PREPRINTS/preprint135.ps.Z`.

[3] Groote, J.F. Hiding propositional constants in BDDs, Logic Group Preprint Series 120, Utrecht University, 1994. Available by ftp from `ftp.phil.ruu.nl` as `logic/PREPRINTS/preprint120.ps.Z`.

[4] Groote, J.F. & Koorn, J.W.C. & Van Vlijmen, S.F.M. The safety guaranteeing system at station Hoorn–Kersenboogerd, in COMPASS'95, pp. 131 to 150, *Proceedings of the 10th IEEE Conf. on Computer Assurance*, Gaithersburg, USA, IEEE, 1995.

[5] Stålmarck, G. A note on the computational complexity of the pure classical implication calculus, *Information Processing Letters*, 1989, **31**, 277–278.

[6] Stålmarck, G. Normalization theorems for full first order classical natural deduction, *Journal of Symbolic Logic*, 1991, **56**, 129–149.

[7] Stålmarck, G. & Säflund, M. Modelling and verifying systems and software in propositional logic, in SAFECOMP'90, pp. 31 to 36, *Proceedings of the 9th Int. Conf. on Computer Safety, Reliability and Security*, Pergamon Press, 1990.

[8] Tip, F. A survey of program slicing techniques. *Journal of Programming Languages*, 1995, **3**, 121–189.