Elastic Train Control: BMC of a Controller for ETCS 3

1 Introduction

This benchmark deals with analyzing the safety of a railway system when operated under a 'moving block' principle of operation. In contrast to conventional interlocking schemes in the railway domain, where static track segments are locked in full, the moving block principle applies headway control as required by the braking distance, reserving a 'moving block' ahead of the train depending on speed and braking capabilities. There are two variants of this principle, namely train separation in relative braking distance, where the spacing of two following trains depends on the current speeds and braking capabilities of *both* trains, and train separation in absolute braking distance, where the distance of two following trains equals the braking distance of the second train plus an additional safety distance (Figure 1). Within this case study we apply the second variant which will also be used in the forthcoming European Train Control System (ETCS) Level 3.

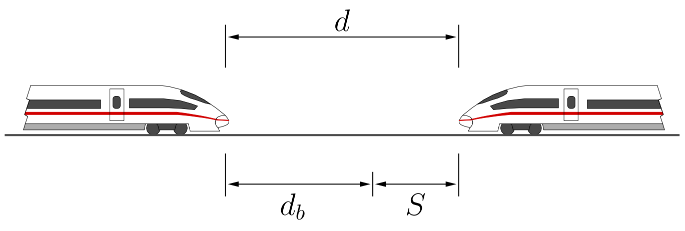

Figure 1: Train separation in absolute braking distance: The distance d between two following trains equals the braking distance d_b of the second train plus an additional safety distance S.

2 Matlab/Simulink Model

We consider an abstract model of ETCS level 3. Within this simplified version, all trains operate in obedience of the following procedures and regulations:

1. All trains on the track travel in the same direction. No train is ever allowed to change direction.
2. There is no possibility of overtaking, i.e. the train sequence is fixed.
3. Each train broadcasts the position of its end to the following train every 8 seconds via radio.
4. Whenever a train receives an update of the position of the train running ahead, it computes
    - its *movement authority* m, i.e. the stopping point it must not cross,
    - and the deceleration a which is required to meet that stopping point.

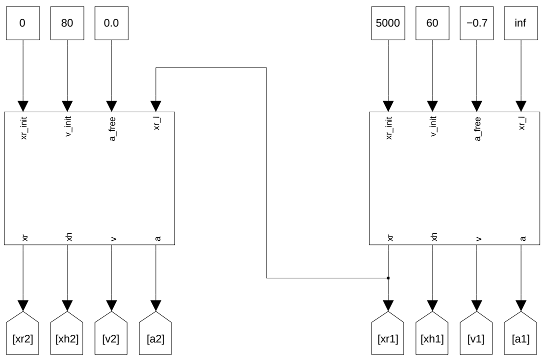

Figure 2: Top-level view of Matlab/Simulink model.

Movement authority and deceleration are computed according to the formulae

$$m = xr - (xh + S) \tag{1}$$

$$a = \frac{v^2}{2m} \tag{2}$$

where xr is the position of the rear of the first train, xh is the position of the head of the second train, and v is its velocity.

Braking is automatically applied whenever the value of a exceeds a certain threshold b_{on}. Automatic braking ends if a falls below b_{off}.

5. When a train is not in automatic braking mode, acceleration and deceleration are freely controlled by the train operator within the physical bounds of the train.

We chose the parameters of the model to roughly match the characteristics of an ICE3 half-train:

Parameter	Value
length of the train $[m]$	200
maximum speed $\left[\frac{m}{s}\right]$	83.4
maximum acceleration $\left[\frac{m}{s^2}\right]$	0.7
maximum deceleration $\left[\frac{m}{s^2}\right]$	-1.4
b_{on} $\left[\frac{m}{s^2}\right]$	-0.7
b_{off} $\left[\frac{m}{s^2}\right]$	-0.3
safety distance S $[m]$	400

Figure 2 shows the top-level view of the Matlab/Simulink implementation of our model in a version with two trains. Inputs of a train block are the inital position of the train, its inital speed, the acceleration applied in free-running mode and the position of the rear end of the train which is running ahead. Outputs are the position of the rear and of the head of the train, its velocity and current acceleration. The implementation of a train block is shown in figure 3.

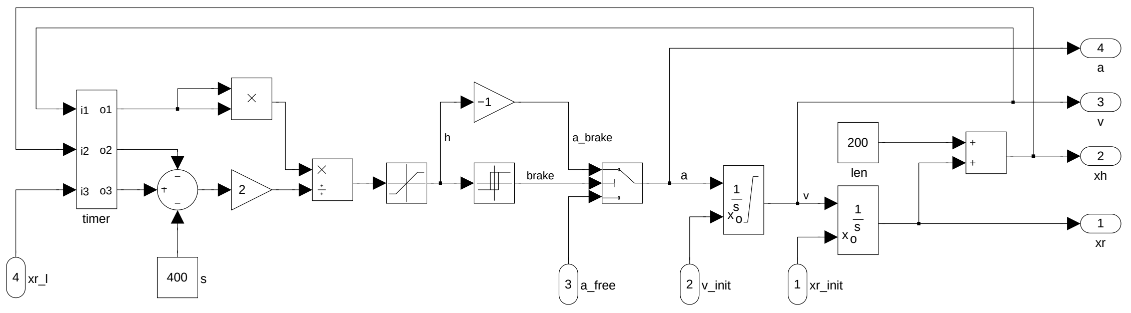

Figure 3: Implementation of controller and train dynamics.

A sample trace of the model, showing position, speed, acceleration and distance of the two trains (see Figure 4), seems to suggest that the controller works correctly: The trains are started with an initial distance of 5000 km, the second train being $20\frac{m}{s}$ faster than the first train, the latter braking with a deceleration of $-0.7\frac{m}{s^2}$. The second train automatically starts braking, adjusting its deceleration in intervals of 8 seconds, and comes to stop exactly 400 meters behind the first train.

Yet, the controller has a simple bug which shall be discovered by bounded model checking: If m becomes zero or even negative (which may happen since the controller re-computes the deceleration setting only every 8 seconds), then instead of applying the maximum braking force, the controller switches back to free-running mode, allowing the operator of the train to accelerate and crash into the rear of the train ahead.

3 Verification Results

To enable bounded model checking with our solver HySAT, we had to manually translate the Simulink model into the input language of our tool. We investigated the following encodings of the model:

- Encoding A with a sampling time of 8 seconds and exact encodings of the integrators.
- Encoding B with a sampling time of 2 seconds and euler approximations of the integrator blocks.
- Encoding C with a sampling time of 1 second and euler approximations of the integrators.

For each encoding we checked whether a collision of the trains is possible if their initial speed is zero and their initial distance is greater than 1000 meters, yielding error traces of length 8 for encoding A, 33 for encoding B, and 66 for encoding C.

The experiments were performed on 2.5 GHz Opteron machine with 4 GByte physical memory, running Linux. The total runtimes for solving all BMC instances up to the error trace were about 10 seconds for encoding A, 1.8 minutes for encoding B and 21.5 minutes for encoding C. The runtime largely depends on the solver settings, e.g. the chosen splitting heuristic. The runtimes reported above are the smallest we could obtain for the respective encoding.

Figure 5 shows an error trace of the model with 66 steps, found by HySAT.

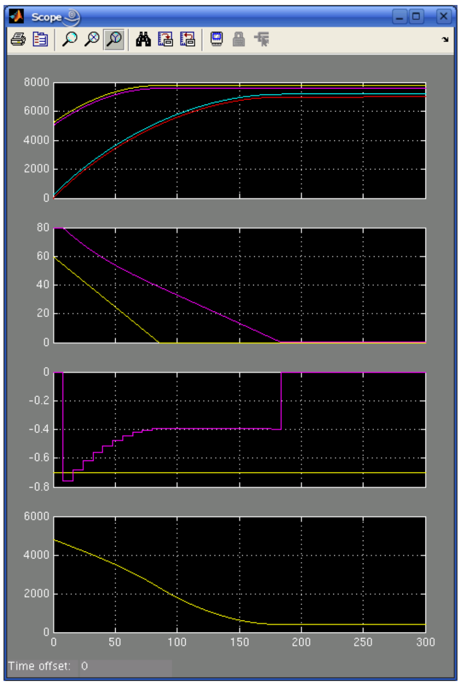

Figure 4: Sample trace obtained by simulation with Matlab/Simulink.

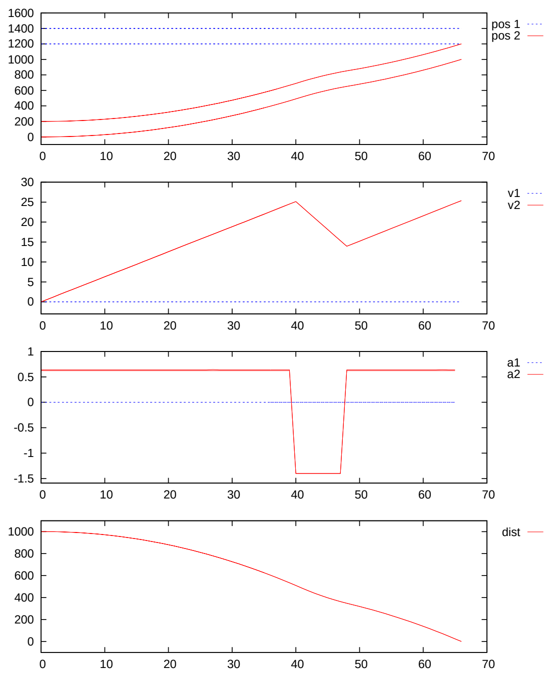

Figure 5: Error trace of length 66 found by HySAT.