Towards safety analysis of ERTMS/ETCS Level 2 in Real-Time Maude

Phillip James, Andrew Lawrence,
Markus Roggenbach, and Monika Seisenberger

Swansea Railway Verification Group,
supported by Siemens Rail Automation

ETWG-RC Amsterdam, 13th November 2015

Overview: ERTMS/ETCS in Real-Time Maude

To investigate how a centralized rail control system,
the European Rail Traffic Management System (ERTMS),
can be modelled and verified using the Real-Time-Maude system

Overview:

- Part I: ERTMS – what it is and how it works
- Part III: Modelling of ERTMS in Real-Time Maude
- Part IV: Validation
- Part V: Verification

ERTMS – what it is and how it works

European Rail Traffic Management System (ERTMS) I

What it is:

- European standard of signalling, control and train protection
- To replace the many incompatible safety systems (20!) currently used by European railways
- Offers possibility for traffic management
- Originally designed for Europe, has rapidly become a global standard.

Some facts:

- Europe: Switzerland (1200km, full coverage by 2017), Denmark (4000km, full coverage by 2024), Germany, Belgium, Spain, Austria; UK: First line in Wales (Cambrian Coast Line, 215km), Norway (80km)
- World wide: China: 8000km, Saudi Arabia, Turkey, 2000km, SA ...

European Rail Traffic Management System (ERTMS) II

Traditional railway interlockings control rail traffic via signals.
In short: ERTMS removes the signals, and replaces them through direct communication between trains and interlockings.

ERTMS shall achieve:

- interoperability
- ease of maintenance (less track equipment)
- higher capacity (40%)

European Rail Traffic Management System (ERTMS) II

Traditional railway interlockings control rail traffic via signals.
In short: ERTMS removes the signals, and replaces them through direct communication between trains and interlockings.

ERTMS shall achieve:

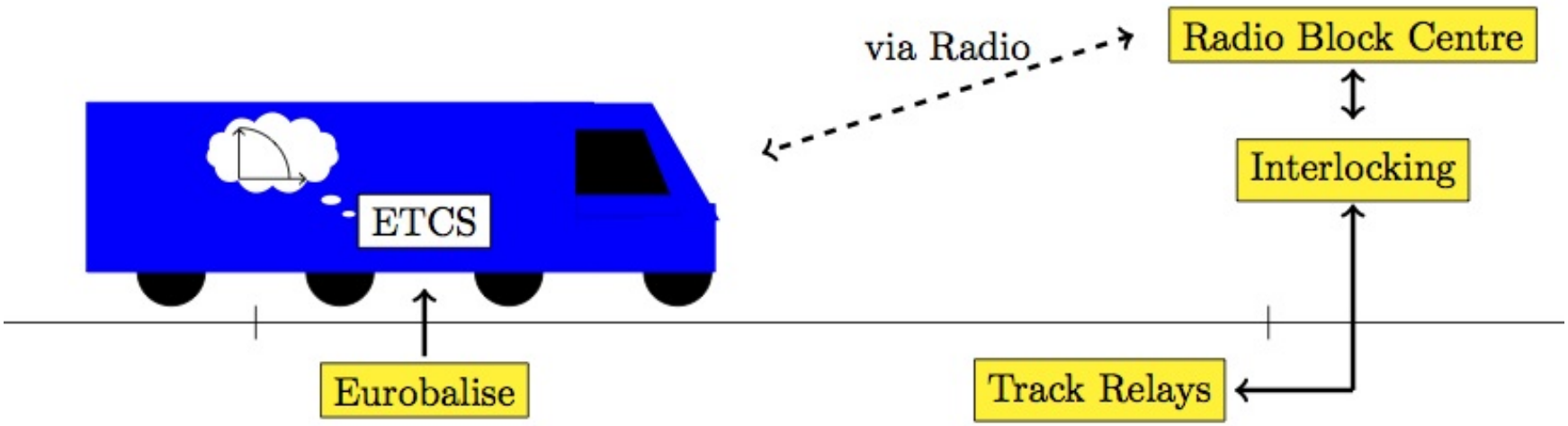
- interoperability
- ease of maintenance (less track equipment)
- higher capacity (40%)

Open research questions include:

1. How can safety be verified?
2. How can capacity be measured and improved?
3. How can reliability be measured and estimated?

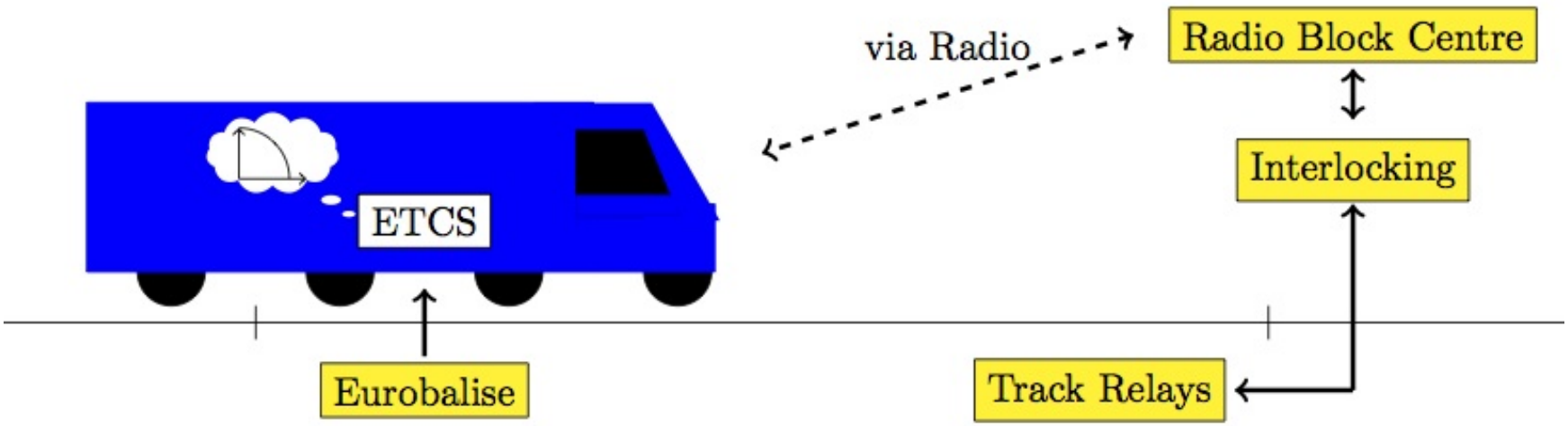
Our work: 1 and, partially, 2.

System components of ERTMS, level 2



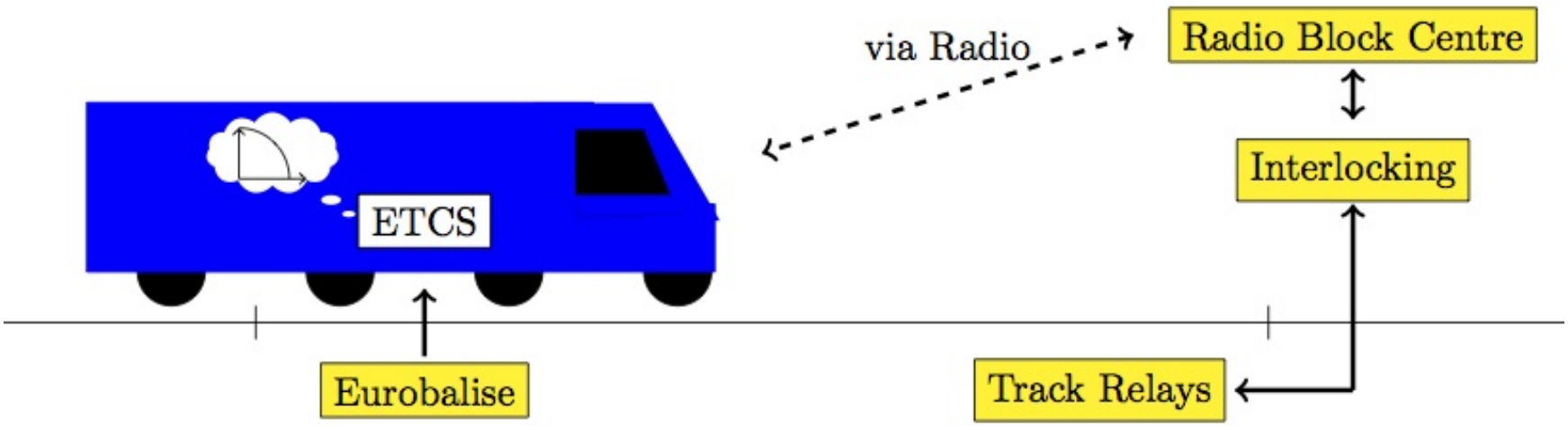
Main Responsibilities:
Trains - communicate position/speed, and receive movement authorities.

System components of ERTMS, level 2



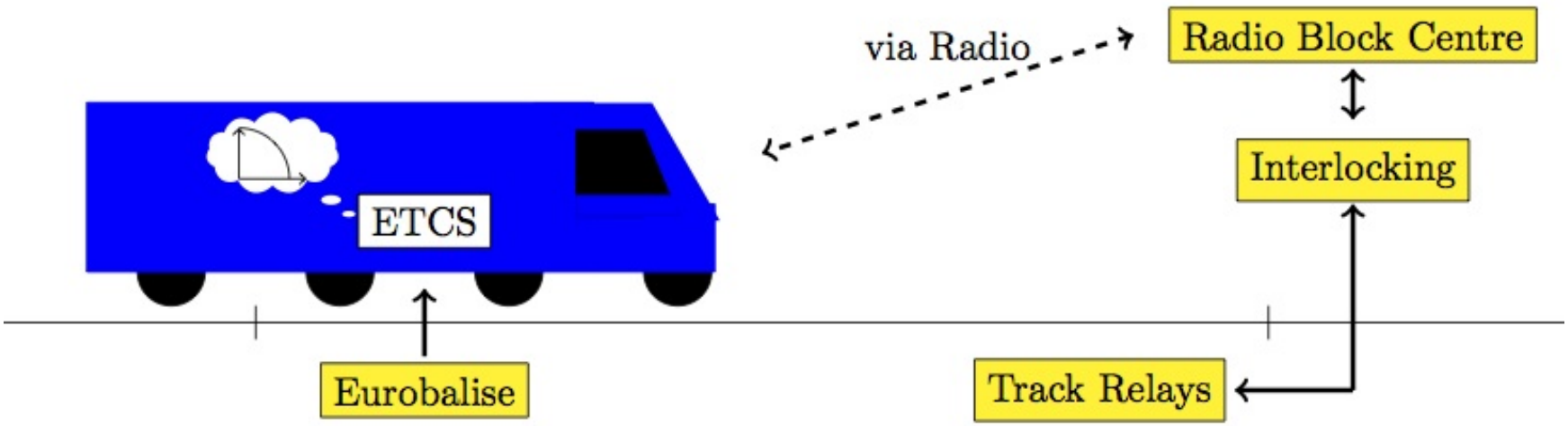
Main Responsibilities:
Trains - communicate position/speed, and receive movement authorities.
RBC - grants MAs/denies MA requests, consults with Interlocking.

System components of ERTMS, level 2



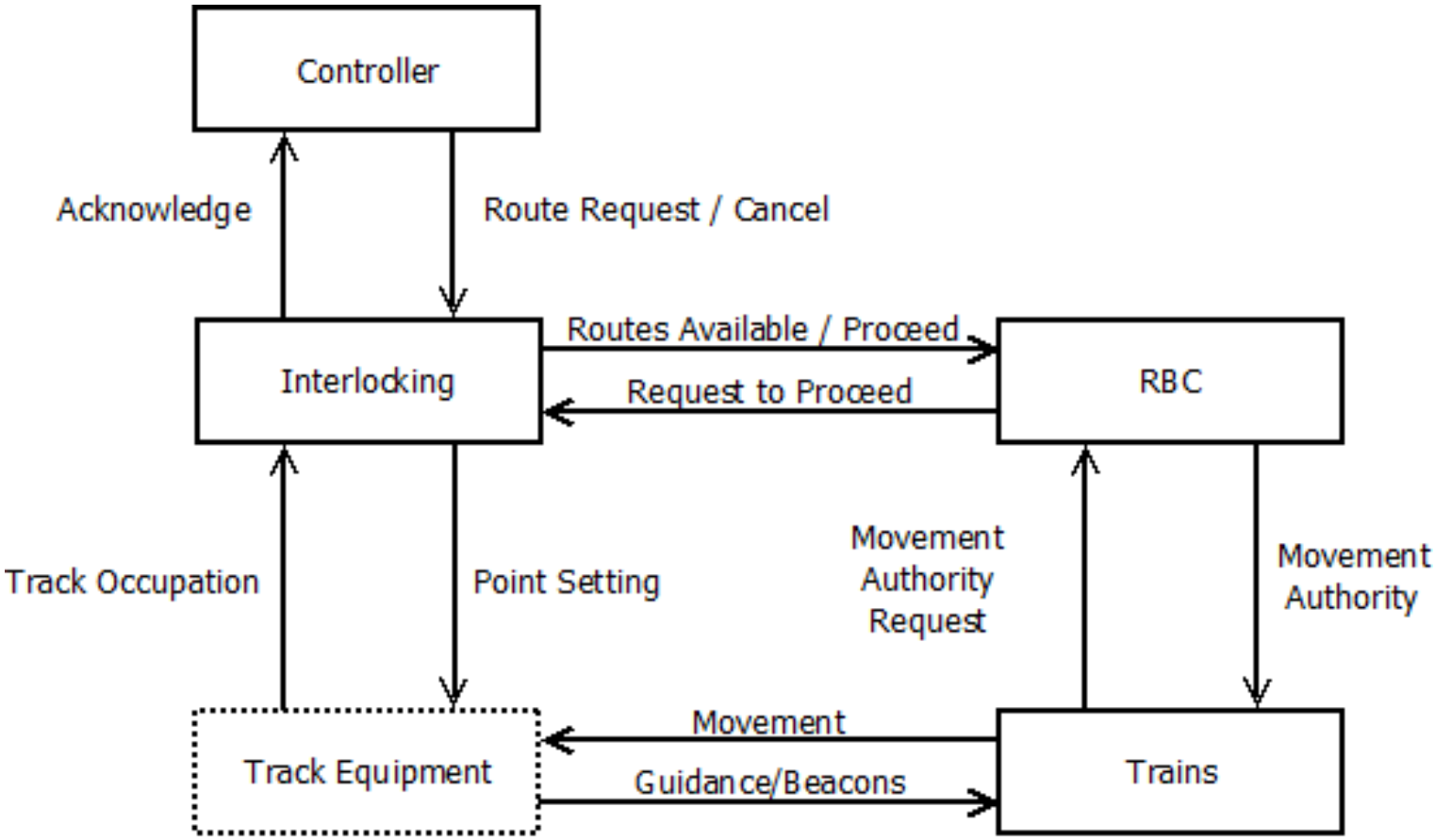
Main Responsibilities:
Trains - communicate position/speed, and receive movement authorities.
RBC - grants MAs/denies MA requests, consults with Interlocking.
Interlocking - allows for setting new routes, responsible for safety.

System components of ERTMS, level 2

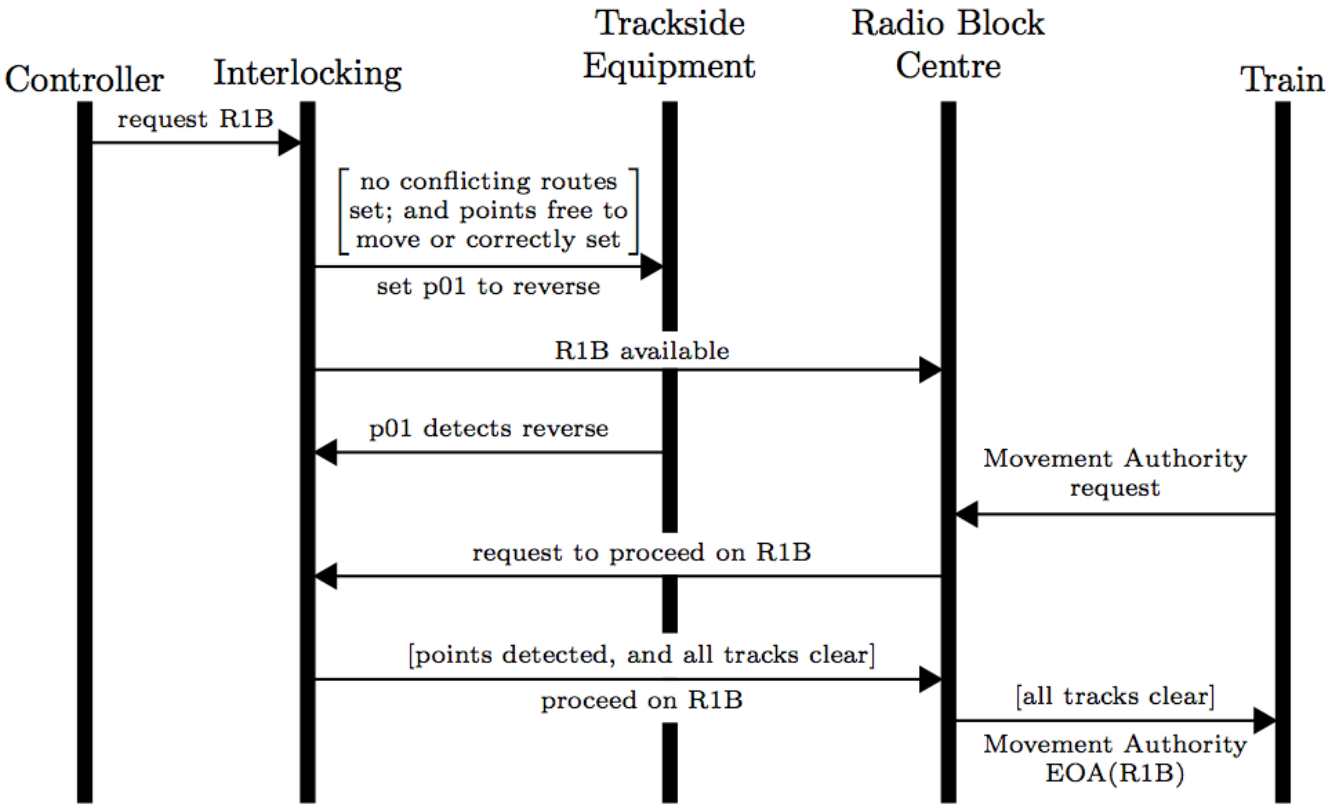


Main Responsibilities:
Trains - communicate position/speed, and receive movement authorities.
RBC - grants MAs/denies MA requests, consults with Interlocking.
Interlocking - allows for setting new routes, responsible for safety.
Controller (not in picture) - requests new routes.

Information flow in ERTMS, level 2



Message exchange in ERTMS, level 2



Modelling of ERTMS in Real-Time-Maude

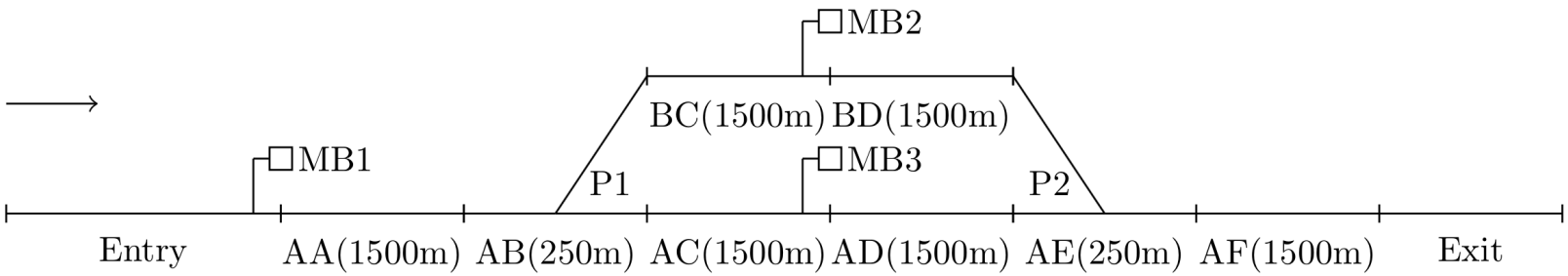
Object Oriented Modelling in Real-Time-Maude

- Real-Time-Maude allows for simulation and formal analysis of real-time and hybrid systems.

- Object based systems are modelled as multisets of objects and messages of a sort `Configuration`, a subset of Maude's built-in in sort System.

- A real-time specification consists of
  - the sort `Time` (in our case `PosRat`),
  - the constructor `{_} : System -> Globalsystem`
  - instantaneous rewrite rules,
  - a so-called tick rule that defines how time elapses.

Scheme Plans



Interlocking Tables:

Route	Clear Tracks	Normal	Reverse
1A	AA, AB, AC	P1	
1B	AA, AB, BC		P1
2	BD, AE, AF		P2
3	AD, AE, AF	P2	

Point	Route	Release
P1	1A	AC
P1	1B	BC
P2	2	AF
P2	3	AF

RBC Tables:

Current Position	Continuation Routes
Before MB1	1A, 1B
Before MB2	2
Before MB3	3

Granted Route	EOA
1A	3249m
1B	3249m
2	6499m
3	6499m

Fig. 1. Scheme Plan for a pass-through station.

Modelling 1: location specific data & messages

Encoding of the rail topology:

```
sort RouteName .  ops RouteName1A ... : -> RouteName .
sort Track .      ops AA AB AC ... : -> Track        .
sort Point .      ops P1 P2 : -> Point               .
```

Messages to be exchanged between the ERMTS components:

```
msg routerequest : RouteName -> Msg .
msg marequest : Oid Track -> Msg .
```

Modelling 2: Instantaneously reacting sub-systems

No time-constraints:

```
eq mte(< O1 : Controller | >) = INF .
```

Modelling 2: Instantaneously reacting sub-systems

No time-constraints:

```
eq mte(< O1 : Controller | >) = INF .
```

Interlocking – a class with internal state:

```
class Inter |  routeset : MapRouteName2Bool,
               pointslocked : MapPoint2Bool,
               occ : MapTrack2Bool,
               pointPositions : MapPoint2PointPos  .
```

Modelling 2: Instantaneously reacting sub-systems

No time-constraints:

```
eq mte(< O1 : Controller | >) = INF .
```

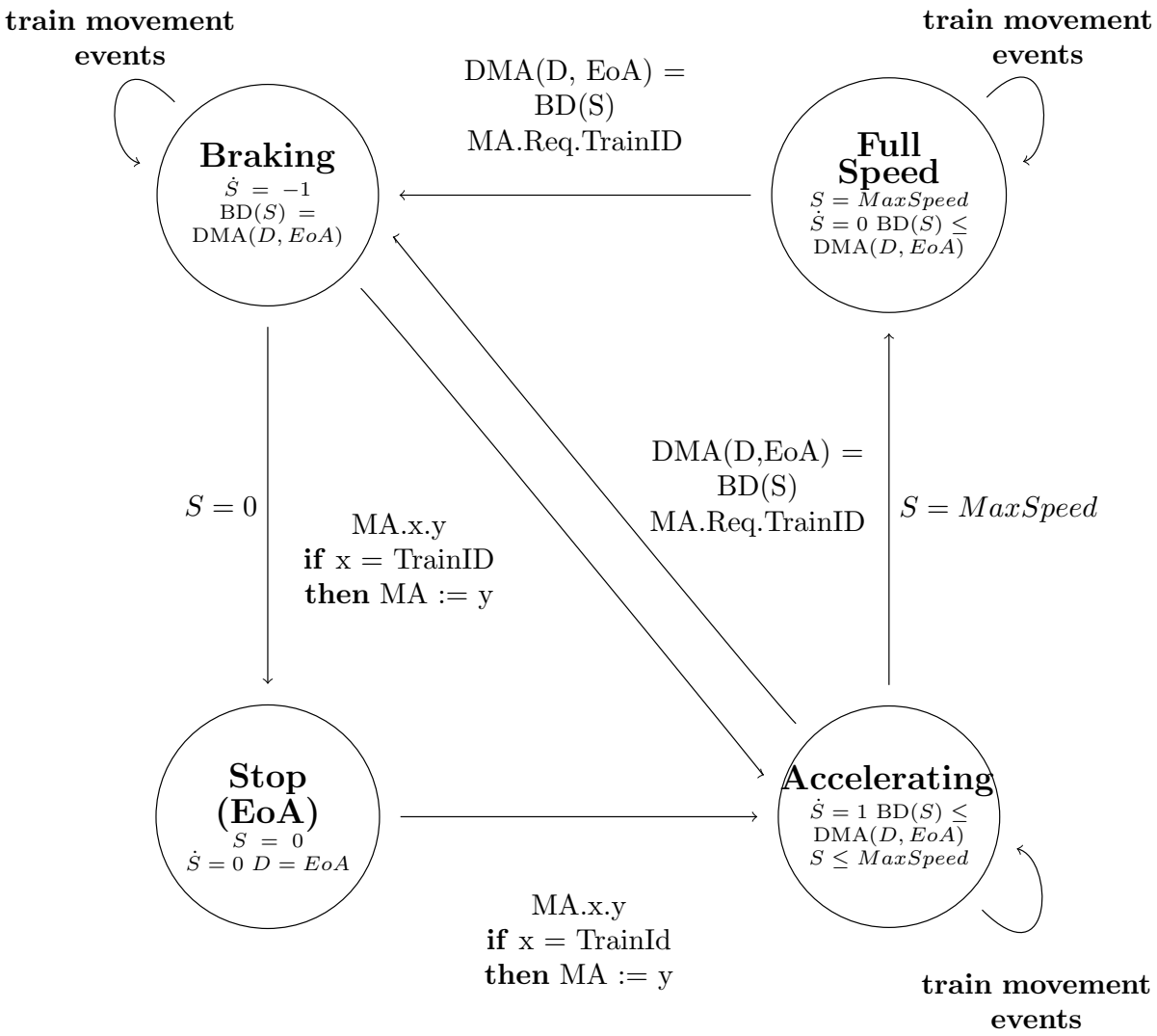
Interlocking – a class with internal state:

```
class Inter |  routeset : MapRouteName2Bool,
               pointslocked : MapPoint2Bool,
               occ : MapTrack2Bool,
               pointPositions : MapPoint2PointPos  .
```

Message processing e.g. ignoring a route request:

```
crl  routerequest(RN1)
     < O : Inter |  occ : MAPTB1, pointslocked : MAPPB3 >
     =>
     < O : Inter | > if (not checkClear(RN1, MAPTB1)) or
                        pointsLocked(RN1, MAPPB3) .
```

Modelling 3: Trains with ERTMS equipment



Modelling 3: Trains with ERTMS equipment

```
crl [acc] :
 ...
 delta(< O : Train | state : acc, dist : DT, speed : S,
          ac : A, ma : MA , tseg : AN, maxspeed : MAX >, R)
 =>
 ...
 < O : Train | state : if (S + R * A == MAX)
                       then cons
                       else (if R == mteMA(DT,S,A,MA)
                               then brake
                               else acc fi) fi,
               dist : DT + S * R + R * R * A * 1/2,
               speed : S + A * R > )
                                    if not AN == Exit .
```

Validation & Verification

Validation through Simulation

We have validated our model through exploring various train movements.

For example, rewriting a train starting on track AA:

```
(trew { < inter1 : Inter | pointPositions : (P1 |-> normal,
P2 |-> normal) , ... > < train1 : Train | state : acc, dist :
2, speed : 0, ac : 1, ma : 1498, tseg : AA , maxspeed : 60 >
} in time <= 39 .)
```

Validation through Simulation

We have validated our model through exploring various train movements.

For example, rewriting a train starting on track AA:

```
(trew { < inter1 : Inter | pointPositions : (P1 |-> normal,
P2 |-> normal) , ... > < train1 : Train | state : acc, dist :
2, speed : 0, ac : 1, ma : 1498, tseg : AA , maxspeed : 60 >
} in time <= 39 .)
```

shows that it accelerates until it is required to begin braking due to its MA:

```
... < train1 : Train | ac : 1, dist : 1499446241/2000000, ma
: 1498, maxspeed : 60, speed : 38671/1000, state : brake,
tseg : AA >... in time 38671/1000
```

Validation through Simulation (2)

It then makes a movement authority request:

```
 marequest(train1,AA) < inter1 :  ...> < train1 :  Train | speed
:  37671/1000, ...  > in time 39671/1000
```

However at this point the system will not progress until we add an RBC to deal with the request... See the paper for the complete example!

Error Injection: RBC Design Error

Our modelling is able to find errors, for example:

Assume the RBC is designed with incorrect EoA values,
e.g. EoA of Route 1A = 3449m

Error Injection: RBC Design Error

Our modelling is able to find errors, for example:

Assume the RBC is designed with incorrect EoA values,
e.g. EoA of Route 1A = 3449m

Model checking is able to produce a couter example where train 1 overuns and hence is able to get within 100m of train 2:

```
 ...< train1 : Train | ac : 1,dist : 3449,ma : 3449,
maxspeed : 20,speed : 0,state : stop,tseg : AD >
 < train2 : Train | ac : 1,dist : 12433788921/4000000,
ma : 6499,maxspeed : 60, speed : 60,state : cons, tseg
: AC > ...
```

Error Injection: Train - Incorrect braking parameters

Another error:

Decelleration used for computation: 1; physical deceleration: 8/10.

```
< train1 : Train | ... dist : 3249, ac : 1, ma : 6499, tseg
: AD , maxspeed : 20 > < train2 : Train | ... ac : 8/10, ma
: 1, tseg : Entry , maxspeed : 60 > ...
```

Error Injection: Train - Incorrect braking parameters

Another error:

Decelleration used for computation: 1; physical deceleration: 8/10.

```
< train1 : Train | ... dist : 3249, ac : 1, ma : 6499, tseg
: AD , maxspeed : 20 > < train2 : Train | ... ac : 8/10, ma
: 1, tseg : Entry , maxspeed : 60 > ...
```

Model checking is able to produce a couter example

```
< train1 : Train | ac : 1,dist : 15662341/2500,ma : 6499,
maxspeed : 20,speed : 20,state : cons,tseg : AF > < train2 :
Train | ac : 4/5,dist : 968593576867/156250000, ma :
7999,maxspeed : 60,speed : 60,state : cons, tseg: AF > ...
```

Safety Verification through Model-checking

Verification that trains cannot be within 100 metres of each other, e.g.:

```
mc initState |=t [] nocrashDistance(train1,train2) .
```

Scheme Plan	**Round Robin Controller Unbounded**	**Random Controller in Time 300**
Junction	2.4s / 5,767,435 rewrites	268.3s / 208,715,358 rewrites
Pass-through Station	3.0s / 7,135,987 rewrites	439.2s / 308,629,500 rewrites
Three Platform Station	2.8s / 6,624,578 rewrites	2697.1s / 729,201,878 rewrites

Table: Verification results of model checking three scheme plans.

Conclusion & Future Work

Summary

The firsts:

- First use of Maude / Real-Time Maude in the railway domain.
- First formal model comprising of all ERTMS subsystems required for the control cycle.

The quality jump:

- Rail control modelled as a hybrid system.
- Safety properties in physical rather than in logical terms.

The verification results:

- Unrestricted controller: time bounded model-checking.
- Restricted controller: unbounded model-checking.

Future Work

Further experiments:

- More, and more complex rail-yards.

Improving the models:

- Bi-directional rail-yards.
- Further controller strategies.
- More complex train progression behaviour.

Reflecting on the models:

- Include further safety properties.
- Address completeness.
- Develop abstractions to increase in verification speed.

Related work

ERTMS is a complex systems of systems – many focus on one subsystem only:

- Vu et al.: Interlocking
- Cimatti et al.: subsystem for allocation of logical routes to trains
- Nardone et al.: radio block centre.

Our objective: verify location specific data – others work towards:

- Software model checking of subsystems.
- System testing.
- Check the consistency of the ERTMS specifications themselves.

We verify system parameters on the *design level* – others focus on verification of the *implementation*.