RTSA

SIGNALLING CONTROL TABLE GENERATION AND VERIFICATION

DAVID TOMBS (SVRC)
NEIL ROBINSON (SVRC)
GEORGE NIKANDROS (QR)

SUMMARY

Queenland Rail (QR) and the Software Verification Research Centre (SVRC) from The University of Queensland are investigating a suite of tools to assist in the production of signalling control tables. Altogether there are four tools, a graphical track layout editor, a tool to generate control tables automatically, a tool to enable manual editing of tables, and a verifier to show that tables satisfy signalling safety principles. This paper provides an overview of the toolset. It gives a fuller description of two of the key parts of the toolset design: the algorithms to generate control table entries and the formalisation of signalling safety principles for verification.

1. INTRODUCTION

Control Tables are the functional specification for railway signalling interlockings. They have an important role in the signalling design process since they act as an agreement between the railway administration and the train operators on when moves will be permitted on a track layout. They also act as a design specification for use by the interlocking designers and a test specification for use by testers. Control Tables contain the key functional safety requirements for the interlocking.

Currently Control Tables are developed and checked entirely manually. QR and SVRC are investigating the automatic generation and verification of Control Tables [15]. In this paper, an overview of a prototype toolset to support Control Table designers and checkers is provided. The paper focuses on two key areas of the toolset: Control Table generation and the specification of signalling principles that will be used for the automatic verifier. The verification itself is the subject of other papers [16] [17].

Section 2 of the paper provides a description of QR's Control Tables. Section 3 gives an overview of the toolset and its architecture. Sections 4 and 5 describe the Control Table generator and the specification of signalling principles.

2. CONTROL TABLES

A *control table* is a structured, tabular presentation of the rules governing route setting on a railway track layout. It is used as a specification of the signal interlocking for the layout and as a test specification for the interlocking. The rules for writing out control tables are derived from the principles of safeworking of trains. A control table represents an intermediate level of design between a track plan and a wiring diagram. The format and content of tables is not standardised, and may vary even within the same railway administration. Diverse safeworking practices and signalling technologies drive diverse table formats. Nevertheless, general principles of control table design are evident. This work specifically addresses the tables and principles applied by QR within the Brisbane Suburban Area (BSA), but may be adapted to other formats. A sample of a BSA control table is shown in Figure 1.

Signal	Route Number	Route to	Route Indication	Requires								
				Points Locked		Signals		Route Holding	or Until		Tracks	
				Normal	Reverse	Normal	Reverse	Maintained by tracks occ	Tracks occ	for Time secs	Clear	Occ
14	2M	16	-	322							14AT 27BT 27AT	
				312				14AT 27BT	27BT	60		
						25		14AT				
						27		14AT 27BT				

Figure 1 - Example Control Table

A *route*, or path from signal to signal, is a key concept in the table. One row of the table indicates the conditions needed to set a single route. Columns of the table indicate the: point settings required by the route; conditions for locking and releasing other routes; tracks required clear in order to clear the entry signal; route replacement rules; and approach locking acquisition and release.

Some signalling conditions are presented in other tables, subsidiary to the main route table. There is a point control table that indicates the conditions needed to move a point. Since the interlocking connects points and routes, the two tables are, to a degree, converses of one another. Other tables cover approach locking, aspect sequencing, level crossings and dual-gauge requirements.

The BSA Principles [1] effectively define the content of each table cell. The task of filling them out is complex, but with experience much of the work is routine. The routine elements are amenable to automation.

However, in general, it does not appear to be feasible to generate a complete table automatically. Some signalling rules are imprecisely defined, some are difficult to capture in the control table format, and at some locations there are special rules that deviate from the general principles. These issues imply a degree of human intervention in the control table generation process.

3. ARCHITECTURE OF THE TOOLSET

The Signalling Design Toolset (SDT) consists of a suite of tools which generates a control table automatically, allows human editing of the table, and verifies the control table against a set of signalling principles. There are four tools altogether, related as shown in Figure 2.

The tools are discussed in turn in the subsections below.

3.1 Track Layout Editor

The Track Layout Editor is used to produce a graphical representation of a track layout. The user draws the track on the screen and decorates it with the elements needed for a control table (track segments, points, signals etc). The editor undertakes consistency checks on the drawing (eg, that precisely three track segments join at a point) and writes out the elements and their connectivity in a formal notation. For reasons of portability and utility we have chosen Extensible Markup Language (XML) as our interchange format.

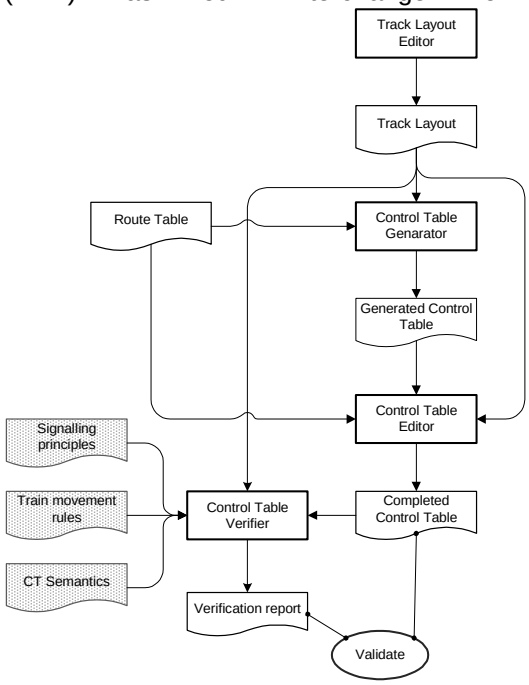

Figure 2 - The Signalling Design Toolset

3.2 Control Table Generator

The Control Table Generator is a tool that generates control table entries for a layout, given its XML description. It generates all entries of the table that can be inferred from the layout. However, in general it is not possible to compute the entire table automatically. Limitations include situations where different options are possible, or where extra commentary or notation is needed to fully specify the meaning of a condition.

The output of the tool is a structured description of signalling controls for the layout. It is expressed in XML, in a form that mirrors the BSA control table structure.

3.3 Control Table Editor

The Control Table Editor allows the user to add, remove or modify entries in a Control Table. Control Tables are presented in a style similar to a spreadsheet. Edited control tables are stored in the same interchange format produced by the Generator.

User editing is essential, for several reasons. First, it is still sometimes desirable to produce a table by hand from scratch. Second, as indicated above, the Control Table Generator will not always produce a complete table. Third, specific local signalling rules apply at

some locations. These local rules may be either more or less restrictive than the general principles.

From the Control Table Editor, the user can perform basic consistency checks on the Control Table. For example there might be a check that each point control in the route table has a corresponding control in the points table.

3.4 Control Table Verifier

The Control Table Verifier is a tool that checks a control table against a set of Signalling Principles. To achieve this it translates a Control Table into a model that defines the behaviour of the signalling objects (e.g. points and signals) for the layout, based on generic Control Table Semantics, which define the meaning of the Control Tables. It then puts this model together with a model that captures assumptions on how trains can move through a layout. This final model is exhaustively checked against the signalling principles, using a model checking algorithm (see Section 5).

We now focus on two key parts of the toolset design: the Control Generation and the specification of Signalling Principles for verification.

4. CONTROL TABLE GENERATION

The BSA principles document acts as a requirement specification on the control table that should be generated. However it does not define an executable algorithm. This section outlines the algorithm used in this work. Different classes of route (eg main and shunt) have slightly different requirements, but the same algorithm captures all cases. The following describes the algorithm applied to a main route.

4.1 The General Algorithm

The algorithm comprises two main phases. Phase 1 identifies the sequence of tracks within each route and the required setting of all points along it. It further finds tracks and point settings up to any overlap limit of the route. It does this by means of a tree walk over the network graph, starting from the route entry signal and continuing until it finds a path to the exit signal.[1]

This phase enables completion of much of the control table, including:

- tracks required clear or occupied, from entry to exit signal;
- overlap tracks from the exit signal to the overlap limit, including all alternative overlap paths (swinging overlaps);
- the required lie of each point lying in the route and overlap;
- in-route shunt signals;
- signal replacement tracks (generally, the first track after the entry signal);
- approach locking release conditions.

Phase 2 finds conflicting routes on the network, ie pairs of routes that cannot be set simultaneously. The control table records conflicting routes in a column indicating which routes are locked normal (ie cannot be set). Two routes that pass along the same track in opposite directions are conflicting; there are other cases as well. With a few exceptions, conflicting route pairs are opposing routes that share any in-route or overlap track. These pairs are extracted from the information gleaned during Phase 1.

Phase 2 also determines the conditions for release of locking of points and conflicting routes. When a train starts to use a route by passing the entry signal, then the locking conditions must not be released until it is ‘safe’ to do so, even though the entry signal is set to a stop aspect. When it has completed the route, then locking conditions may be released[2], and the route is now ‘normal’ once more. Locking conditions may be released progressively as the train passes along the route to allow better utilisation of the rail asset and operational flexibility.

To understand the algorithm, consider a train passing along a route, from entry to exit. As it passes each trailing or facing point, then the reason for locking the point no longer applies. Similarly, as it passes an opposing signal, then any conflicting route starting at the signal may now be set. Finally, in-route shunts are released as the train passes the shunt signal. The train effectively maintains the locking on each point and route that requires a track in front of the train, but not those only requiring track behind. Tracks whose occupancy

[1] The present algorithm assumes there is only one route from entry to exit. Multiple routes must be handled manually in the Control Table Editor.

[2] Of course, as a train completes one route and starts another, it maintains locking on an opposing route in conflict with both.

maintains the locking are known as the *holding tracks* of the point, conflicting route or in-route shunt. As expressed above, the complete set of conflicting routes and holding tracks contains a degree of redundancy. Some routes may be omitted from the control table and interlocking logic because they are indirectly locked normal by other conditions in the table. Routes that require a point setting different from a route in use are locked by the point. In-route shunts lock opposing routes normal, on behalf of main routes using them. Hence the control table can be optimised by removal of redundant route controls and holding conditions. The Phase 2 algorithm performs these optimisations.

4.2 An Example

To illustrate the algorithm, consider the layout in Figure 3.

One route passes left-to-right, from signal s1 to s7. There are four potentially conflicting routes passing right-to-left, all starting and terminating at an even-numbered signal. The example is simplified for explanatory reasons. We assume that all routes are main routes, there are no in-route or other shunts, and we ignore overlaps and timed release of locking.

Applying Phase 1 to all routes identifies the in-route tracks and point settings for each route. The results are shown below:

route	entry	exit	Tracks	points set
r1(1m)	s1	s7	1A, 1B, 1C, 1D, 1E, 1F	p101 N, p102 N
r4(1m)	s4	s2	4A, 1B, 1A, 4B	p101 R
r6(1m)	s6	s2	1C, 1B, 1A, 4B	p101 N
r10(1m)	s10	s8	10A, 1F, 1E, 10B	p102 R
r10(2m)	s10	s6	10A, 1F, 1E, 1D	p102 N

The route labels *r1(1m)* etc are synthetic, but are derived from QR practise. A suffix *N* on a point indicates a normal setting while *R* indicates a reverse setting.

Now consider application of Phase 2 to the route *r1(1m)*. All four even-numbered routes are conflicting, directly or indirectly.

The holding tracks for the two points are obvious. If a train has just entered *r1(1m)* (ie, it occupies track 1A) then moving either point might divert it along the wrong path or cause a derailment. However it is safe to move *p101* once the train has passed 1B and to move *p102* once it has passed 1E. Route *r6(1m)* opposes *r1(1m)* and is directly locked because both routes require the shared point *p101* normal. Therefore it must appear in the 'routes locked' column for *r1(1m)*. The locking is held until the train passes *s6*. Note that *r6(1m)* will also be locked normal by the route applying up to *r1(1m)*.

Route *r4(1m)* opposes *r1(1m)* but is indirectly locked via the setting of facing point *p101*. The locking may be released once the train clears *p101*. Therefore the point locking rules are sufficient and *r4(1m)* need not appear in the 'routes locked' column for *r1(1m)*.

When considering route *r10(1m)*, point locking rules are not sufficient in this instance (r*1(1m)* requires *p102 N* whereas *r10(1m)* requires *p102 R*) for when the train has cleared *p102* (i.e. *1E* track clear, *1F* track occupied) *p102* could be swung behind the train. This should not release the locking of *r10(1m)* as it is now directly opposing the train movement. Route holding would list all tracks up to and including the point tracks after the points.

Route *r10(1m)* opposes *r1(1m)* but is indirectly locked via the setting of point *p102*. However, because *p102* is a trailing point from the viewpoint of the train, the locking cannot be released once the point is cleared, unlike *r4(1m)*. 1F is a holding track of the route. In theory, holding tracks 1A to 1E are indirectly locked by *p102* and need not appear in the table next to *r10(1m)* but QR practise is to list all holding tracks in this situation.

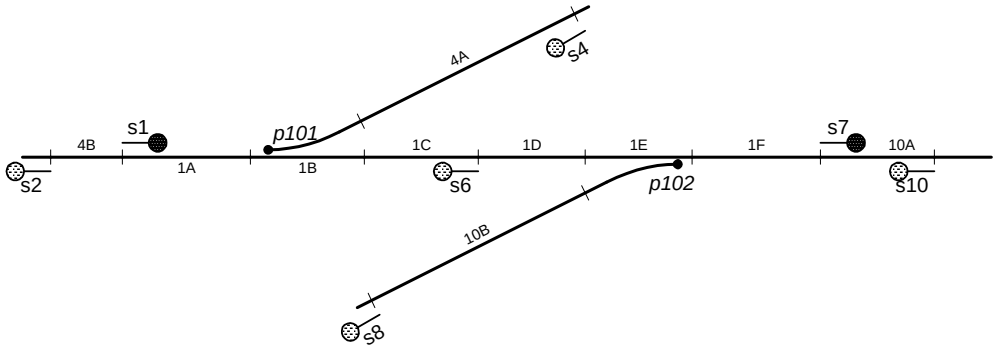

Figure 3 - Example track layout

Therefore, the generated table for *r1(1m)* after Phase 2 looks like:

route	entry	exit	tracks	points locked	routes locked	holding tracks
r1(1m)	s1	s7	1A, 1B, 1C, 1D, 1E, 1F	p101 N		1A, 1B
				p102 N		1A, 1B, 1C, 1D, 1E
					r6(1m)	1A, 1B, 1C
					r10(2m)	1A, 1B, 1C, 1D, 1E, 1F
					r10(1m)	1A, 1B, 1C, 1D, 1E, 1F

Table 1 Partial control table after Phase 2

4.3 Limitations of the Algorithm

Currently the generation algorithm does not generate the controls for swinging overlaps. Instead it merely records facing points in the overlap that trigger swinging overlaps. Also, the algorithm does not handle multiple routes from entry to exit.

4.4 Prototyping

There are two prototypes of the Control Table Generator.

The algorithm of section 4.1 was developed during an initial prototyping phase, using a specification and animation language. The Possum animator [9], based on the Z specification language, was chosen. This was in order to provide the benefits of formal specification, with the additional benefit of execution of the specification. The animation was developed to the point where it could handle a single loop station and its approaches, including main and shunt routes and locking and release of opposing routes and in-route shunts. However, the Possum prototype has a number of drawbacks that mean it is not suitable as a final prototype. Possum demands a certain style of specification in order to work efficiently. Operations need to be broken up into small chunks, and the specification needs to be fairly explicit (Z allows implicit specifications). The specification needed to be altered to make it work efficiently in the animator, and this made it less elegant. Even with the alterations, the animation was still slow. For example the animation took half an hour to generate the control tables for a simple passing loop station.

The second prototype, currently under development, attempts to remedy these defects. It is written in an imperative programming language (Ada) and copies the algorithm from the Possum prototype. It performs full input and output processing via XML and covers both route and point tables. When complete, it should be significantly faster to execute and handle large, real-world layouts. It should be sufficient to act as a prototype for a full operational system.

4.5 Related Work

The terminology of *direct* and *indirect* locking of opposing routes is taken from Hachiga [8], who specifies an algorithm for identifying opposing routes and caters for complex conditions such as flank protection.

Cullyer & Wang [5] use the HOL specification language to describe a track layout and to formalise signalling principles, at a level comparable with the BSA Principles. They present an overview of an implementation in SPARK Ada.

Brown & Haberlin [4] demonstrate how a computer program may be used to assist in the automation of control table generation on old BR Southern region. There is an informal definition of such terms as route holding, aspect sequencing, and approach locking release. The tool was tried in a number of real interlocking designs, and a claim is made that it revealed previously unknown errors and improved process efficiency.

5. SPECIFICATION OF SIGNALLING PRINCIPLES

Automatic verification of Control Tables is one of the key functions of the Signalling Design Toolset. In the prototype toolset, the automatic verification is performed using the *model checking* tool SMV, which checks a specific Control Table against a set of properties. A key problem with model checking is the *state explosion* problem. As the size of the model to be checked increases, in general the time and/or memory requirements increase exponentially [16]. The properties that are checked must somehow capture QR's signalling principles. It is the formal specification of these properties that are the focus of this section.

There are different levels of abstraction at which signalling principles may be defined or formalised. If the principles are defined at a low level, it can be expected that the definition will be large, complex, hard to capture in a formal notation, and hard to validate against intent. However, it should be relatively easy to formally verify a control table or implementation against low-level principles, although the control tables may be so close to the principles that verification may become trivial and meaningless. Conversely, a high-level definition of principles should be smaller, easier to formalise, and more obviously correct with respect to intent, but the verification task will be harder, at least in terms of logical complexity.

5.1 Survey of existing approaches to specifying signalling principles

Most railway administrations, including QR, define their signalling principles in informal natural language descriptions. These descriptions can become long and complex. This is a particular problem in the UK, where they define their principles in a similar manner to QR. The Institution of Railway Signalling Engineers (IRSE) in the UK has conducted a review of Signalling Philosophy [10], which included a report from a working group on Signalling Principles. In the IRSE's report, they state that "Improvements are needed to the body of existing UK signalling principles…. Existing signalling principles are not sufficiently complete, nor are they written in a formal manner so as to be a generic specification for signalling systems,…They also lack explanation of the underlying assumptions and rationale, and therefore, depend heavily on the knowledge and experience of designers and testers."

There is therefore recognition that signalling principles should be specified in a more formal manner, and with clear traceability to underlying assumptions and rationale. Most approaches to re-specifying signalling principles have focussed on the formalisation aspects.

For example, Eriksson [6], [7] has undertaken a substantial formal description and verification of railway interlockings on behalf of Banverket (Swedish National Rail Administration). He verifies a model of the interlocking circuitry (relays, etc) (implementation model) against a formalisation of certain signalling principles (specification model). The principles are expressed at a comparable level to QR's BSA principles; for example, there is a rule that points in a route must be set in the correct direction.

Borälv [3] and Petersen [13] describe interlockings in a special purpose language called STERNOL, and verify them using NP Tools. The work is at the same level of abstraction as Eriksson's work.

Praxis Critical Systems [14] have completed a specification of interlocking requirements for British Rail. The specification captures the requirements in structured language, supported by a Z specification. The objective of the work is to have a rigorous specification to give to interlocking developers.

Some other work has attempted to address the problem of traceability to assumptions and rationale. For example, Lemos, Saeed and Andersen [12] develop a collisions-based specification of a railway network, expressed using a logical formalism (Timed History logic) and a Petri transition network. The

specification states an invariant that two trains must not lie at an intersection (must not collide), develops a strategy for train movement that allows them to proceed, and shows that the strategy meets the invariant. However, their method is applied to a "toy" train set comprising two intersecting circles, and there is no indication of how to apply it to more realistic layouts.

A different approach was taken on Railtrack's West Coast Main Line project in the UK. Here they derived a collection of abstract signalling rules, based on a set of distinct operational 'scenes'. For each scene, accident sequences, safety requirements and operational objectives are elaborated. An example is a scene with two trains following on plain line, where an accident sequence is a rear-end collision due the leading train slowing down. The scenes are written in structured natural language. The objective of the work is to make a high-level statement of principles for transmission-based signalling principles, but the work also applies to fixed block signalling.

5.2 QR's existing specifications of Signalling Principles

The specified principles must be consistent with QR Signalling Principles. QR maintain two relevant documents - the Brisbane Suburban Area (BSA) principles [1] and the more general "Principles of the Signalling of Trains" [2].

5.3 BSA Principles

The BSA principles document is at a relatively low level of abstraction, and closely corresponds to the BSA control tables. For example, the Section "Main route general requirements" lists a collection of items that are clearly identifiable with specific control table columns, eg "(b) opposing and conflicting routes normal", "(g) in-route tracks clear" and "(h) tracks in overlap clear". The meanings of these items are defined in more detail, but still rather informally, elsewhere in the document. In part, the document acts as a manual for completing control tables.

The BSA Principles document also contains a lot of material concerned with implementation of the rules, which have little direct impact on the control tables or their correctness, eg proving electrical track circuitry. This material does not impact on the formal verification task, and therefore not on the definition of formal principles. The BSA principles document does not present a rationale for the signalling principles.

Initially, the project considered using the BSA Principles document as the main source of principles for verification. The real effort appeared to be in formalising the BSA principles. Once this is done, then verification of control tables is relatively straightforward. However the set of formalised principles would be very hard to validate, because of their size and complexity. Errors in the formalisation could lead to errors in the verification of the control tables. Also, since the control tables and the Control Table generation algorithms are so closely related to the BSA principles, it is likely that the errors made in the generation side would be the same as the errors made in the verification side - a potential common-mode failure.

5.4 Principles for the Signalling of Trains

QR also maintain a more abstract document, the "Principles for the Signalling of Trains". This document defines general signalling principles that apply to all signalling systems used on the QR network, not just Brisbane Suburban Area. It is not restricted to fixed-block, trackside signals, and allows for in-cab signalling and Automatic Train Protection (ATP).

The document states a number of high level requirements, eg:

- a safe distance between trains must be maintained;
- the integrity of a route must be maintained, once a proceed authority has been given;
- guidance and braking and overlap distances.

Because this document is more abstract than BSA principles, it would seem to be a more appropriate source for formalisation. However, the generalisation comes through terminology applicable to different signalling systems, rather than a more abstract form of the principles appropriate to BSA territory. For the project's purposes, that generality is hard to sustain when verifying a particular layout.

As an example, consider 5.1, "Train Separation".

> *A safe separation shall be assured by either: maintaining at least one unoccupied intervening Route; or providing an overlap limit beyond the end of authority; or by controlling the entry speed of the Train into*

a Route, where a full overlap is not available.

For BSA territory, concepts like “route” and “end of authority” must be expressed in terms of the track objects of the layout. In turn, the track objects (track circuits, signal aspects, etc) are driven by the particular signalling rules applying to the layout. For example, main routes in BSA territory meet the requirement by providing an overlap limit according to 3.3.1(h). Therefore, a formal definition based on the above principle will run into the same problems as one based on BSA Principles.

5.5 The Chosen Approach

The project’s chosen approach is to define a set of signalling principles, consisting of:

- operational objectives that the signalling system is intended to support, e.g., the need to join and split trains;
- accidents that the signalling system is intended to prevent, e.g., head-on collision of trains and;
- assumptions on the behaviour of trains/drivers, e.g., that trains do not exceed the limits of their authority.

For the purposes of control table verification, the most important aspects are the accidents and the assumptions on train/driver behaviour. This is because the highest priority is to check the safety of the control tables.

The next Section contains an informal description of these principles. The research project includes investigation of a range of different formal verification technologies, each of which has its own associated formalism. The principles described here can be readily translated into an appropriate formalism. For example, some of the principles have been translated into Computation Tree Logic (CTL), an input language for the Symbolic Model Verifier tool (SMV), and the assumptions on train behaviour have been translated into an Abstract State Machine (ASM), which is automatically translated into SMV code. This is the subject of a separate paper [16].

Only principles that are appropriate to the Brisbane Suburban Area are considered. Note that both the interpretation of each of the accidents and the assumptions on train behaviour are specific to the signalling technology, e.g., lineside signals, and the signalling philosophy, e.g. route signalling, used in the Brisbane Suburban Area.

The accidents listed here are the ones relevant to verifying control tables. At this stage of the project, these principles are still draft. Examples are provided here only to demonstrate the approach to their specification, and are not necessarily complete or correct.

5.6 Accidents - Examples

- *Trains collide at speed*

Interpretation: Two trains on the same track section, at least one of them travelling on a main route and not stopped.

- *Train collides with fixed obstruction*

Interpretation: A train travelling on a track section containing a fixed obstruction, e.g., a buffer stop, having received a more permissive aspect than it should have.

- *Train collides with road user at level crossing*

Interpretation: A train travelling on a track section containing a level crossing, with insufficient or no warning, or with boom gates open or not closed for long enough.

- *Train derailment due to movement into out-of-gauge zone*

Interpretation: Train enters a track section with a different gauge to the train gauge.

- *Train derailment due to excess speed at turnout*

Interpretation: Train passes over facing points having received a more permissive aspect than it should have.

- *Train derailment due to passage over trailing points wrongly set*

Interpretation: A train enters a track section containing a trailing point, where the point is not set and detected in the position appropriate for the train’s movement.

- *Train derailment due to points driven beneath train*

Interpretation: Points move when a train is occupying a track that contains the points.

5.7 Assumptions About Train Behaviour - Examples

The assumptions made about train behaviour govern the effectiveness of the control table verification. The more sophisticated the

assumptions, the more of the control table can be checked, but the more difficult the verification becomes.

Currently, the prototype verifier uses a very simple model of train behaviour which is sufficient to check only the most critical parts of the control table, e.g. opposing locking between signals and point locking. In this model, trains have infinite deceleration, i.e., if a train meets a red signal it always stops, regardless of the aspect it received at the previous signal. With this model it is not possible, for example, to check the approach locking controls in the control table or the aspect controls, since a train approaching a signal will always stop if it is red.

Some example assumptions from our simplest train movement model are:

- *Trains move contiguously from track section to track section;*
- *Trains only enter or exit an area from the boundaries of the area, or from a non-track-circuited siding.*
- *Trains only enter the area in a manner consistent with the control tables for the adjacent area*
- *Trains continue in the same direction, except when signalled in the opposite direction when they may reverse;*
- *Trains are always detected by track circuits while in a section*[3]*;*
- *Trains always stop at red signals*
- *Trains travel at the appropriate speed for their route, as indicated by the previous route indication*

There are also assumptions on the behaviour of signalling objects, for example that points always move into their controlled position.

The project has also considered a more realistic train model in which trains only stop at signals when they have been given sufficient warning, and in which trains can travel into their overlap. However, currently the feasibility of model checking control tables using these assumptions has not been investigated.

[3] Sometimes, short-term transients in the track circuit mean that a real train is not detected. However the rules of the control table are not designed to cover such situations. We assume that the track circuit gives a continuous reading.

5.8 Handling Local Principles

Local principles are special signalling rules that apply only at specific locations. They may be more restrictive than the general principles, or more permissive than the general principles. More restrictive rules are consistent with the principles; less restrictive are potentially inconsistent, and may provoke a safety violation.

The project investigated an approach in which local principles were formally specified and merged with the general principles before being checked in the Control Table verifier. However, this was found to be overly complex. The project is now following an approach in which local principles are only specified informally, and implemented manually in the control tables. With this approach, the control table verification will have one of two outcomes.

- If the local principles are more restrictive than (consistent with) the general principles, then the verification will succeed. This is correct, because no safety principle is violated by adding extra controls.
- If the local rules are more permissive (inconsistent) than the general principles, an error will be reported by the verifier. This is correct, because a safety principle is violated by removing a control.

In the second scenario, it is for the QR management process to decide whether to accept the risk and proceed with interlocking design, or change the Control Tables or track layout.

5.9 Remaining Issues

It turns out that using the approach with a model checking tool does not detect all relevant errors in a control table. This is mainly because of redundancy in the control table controls. For example, errors in a control table can be introduced, which enable two opposing signals to show green. However, because of the controls on routes leading up to those signals, it is not possible to move trains into a position where they would collide, and so the model checker does not detect an error.

For now, the project has decided to live with this limitation. The verification approach detects the most critical errors – ones which would lead to accidents in practice. In the future the verification against the general

safety principles will be complemented with a less critical set of syntactic and consistency checks (implemented in the Control Table Editor). The intention is to check the consistency of parts of the tables that are known to be redundant.

6. CONCLUSIONS

An approach has been described to generating control tables, that has been shown through prototyping to be capable of generating the majority of the controls in the QR control tables.

A means of specifying the signalling principles has been defined, which, because of their expression in terms of high level accidents and assumptions on train movement, are relatively straightforward to validate. Work with the model checking tool SMV [16] indicates that these principles are appropriate for the verification activity and achieve the most important aim – that of checking the critical safety properties for a control table.

Currently the project is continuing with prototyping the control table verifier, and is developing design specifications for the other parts of the toolset. Development is expected to start mid 2002.

7. ACKNOWLEDGEMENTS

We would like to thank David Barney, David Haley, Kirsten Winter and Peter Kearney for their assistance in developing the ideas in this paper.

8. REFERENCES

[1] Signalling Principles - Brisbane Suburban Area, SAOS Standards, QR, version 2D, 10 August 1999.

[2] Principles for the Signalling of Trains, Safety Management System, QR, version 1.0, 19 January 1998.

[3] A Borälv. Case Study: Formal Verification of a Computerised Railway Interlocking. Formal Aspects of Computing, vol 10, pages 338-360, 1998.

[4] V Brown & A H Haberlin. Computer Aided Production of Signalling Control Tables. IEE Technical Meeting, March 1978.

[5] J. Cullyer, Wai Wong. Application of formal methods to railway signalling - a case study. In Computing and Control Engineering Journal, 4(1): 15-22. 1993.

[6] L-H Eriksson. Formalising Railway Interlocking Requirements. Swedish National Rail Administration Technical Report 1997:3, 1997

[7] L-H Eriksson. Formal Verification of Railway Interlockings. Swedish National Rail Administration Technical Report 1997:4, 1997

[8] A Hachiga. An algorithmic approach to the verification of a railway interlocking table. *International Conference on Computer-Aided Design, Manufacture and Operation in Railways*, 1996

[9] D. Hazel, P. Strooper and O. Traynor, *Possum: An Animator for the SUM Specification Language*, Proceedings Asia-Pacific Software Engineering Conference and International Computer Science Conference, IEEE Computer Society Press, December 1997

[10] IRSE Signalling Philosophy Review – April 2001, Executive Summary, The Institution of Railway Signalling Engineers, Savoy Hill House, Savoy Hill, London

[11] K. M. Hansen. Validation of a Railway Interlocking Model. In M. Naftalin, T. Denvir, M. Bertan, editors, *Procs of {FME}'94: Industrial Benefit of Formal Methods*. Volume 873 of LNCS, pages 582--601. Springer-Verlag, 1994.

[12] R de Lemos, A Saeed & T Anderson. *A Train Set as a Case Study for the Requirments Analysis of Safety Critical Systems.* The Computer Journal, Vol 35, No1, 1992.

[13] J L Petersen, Formal Requirement Verification of a Swedish Railway Interlocking System. Technical Report, Technical University of Denmark, September 1996.

[14] Praxis Critical Systems. Specifying British Rail's Signalling Rules. Technical Meeting of the *Institution of Electrical Engineers*, London, 16th January 1997.

[15] N. Robinson, D. Barney, P. Kearney, G. Nikandros, D. Tombs. Automatic Generation and Verification of Design Specifications. SVRC, The University of Queensland, Australia, TR 00-39, December 2000.

[16] Kirsten Winter and Neil Robinson, *Modelling Large Railway Interlockings and Model Checking Small Ones*, to be submitted to the International Conference on Formal Engineering Methods, ICFEM 2002

[17] Kirsten Winter, *Model Checking Railway Interlocking Systems*, Proceedings of the Australian Computer Science Conference (ACSC 2002), Australian Computer Science Communications, Volume 24, Number 1.