Modelling British Rail's Interlocking Logic: Geographic Data Correctness

M. J. Morley
Laboratory for Foundations of Computer Science
Department of Computer Science, University of Edinburgh
Edinburgh EH9 3JZ.

e-mail: `mjm@lfcs.ed.ac.uk`

November 1991*
(Revision: July 10, 1993)

Abstract

This paper examines the integrity of a portion of British Railways' signal control logic as expressed in their *Geographic Data Language* [Bri88, Mit90]. The issue addressed here is the correctness of the Geographic Data describing a simple signalling system, where this is motivated by the need for a proof tool to assist signalling engineers in verifying their designs. A translation of the Geographic Data Language into CCS is given which forms the first step of a rigorous demonstration that the model developed has certain desirable safety properties.

1 Introduction

In over a hundred years of practical experience, railway signal engineers have achieved very high levels of integrity in the Interlocking systems that have been developed. Traditionally these systems were electro-mechanically controlled but, with an increasing need for greater cost-effectiveness and flexibility, British Rail have begun to introduce computerised control systems. A Solid State Interlocking (SSI) is such a system and has embedded within it a model of the portion of the network it controls. Our interest here is in the Interlocking logic which guides the safe operation of the SSI. This 'logic' is expressed as a collection of operating rules in a geographic database, set out in a notation called the Geographic Data Language. However, these data are manually compiled and, as this process is tedious and prone to error, they must be carefully examined

*LFCS report series: ECS-LFCS-91-186

before any confidence can be gained in the SSI as a control system. Ideally, it should be proven formally that the data contain no inconsistencies which could lead to dangerous conditions in the network.

By representing the Interlocking as a finite state machine it has been found that even for small signalling systems the number of states to examine is enormous. However, CCS [Mil90] has proved to be a convenient formalism for expressing the data correctness problem concisely. In the space of a few pages it is not possible to describe the SSI nor the Geographic Data Language in great detail, although enough of the basic ideas are introduced for this account to stand alone. Further discussion may be found in, for example, [Bri88, Mor91, Cri87, AC91].

2 The Solid State Interlocking

The overall function of the SSI is to adjust the settings of signals and points so as to permit the safe passage of trains. The physical coupling between the SSI and the railway network is achieved through a high speed data highway connecting various track-side modules. These hardware devices relay status information about the network back to the Interlocking and effect the change in state of the signals and points. The Interlocking polls each track-side module cyclically, in the process issuing commands to the network and updating its internal state. Figure 1 reproduces a small section of the signalling

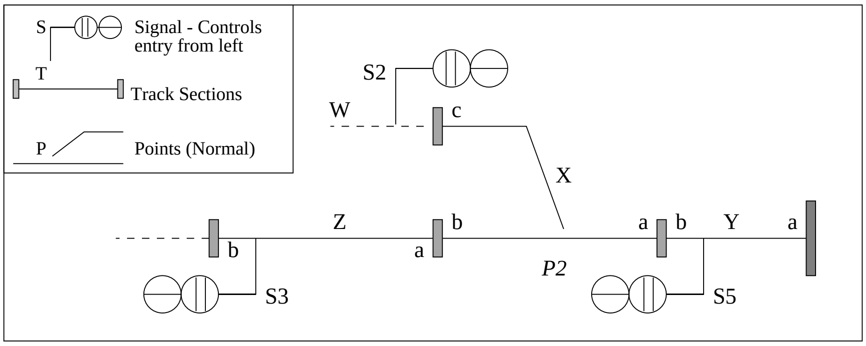

Figure 1: *Network layout for route R_{53}: from signal S_5 to signal S_3.*

system specified in [Bri88, Mit90], which provides the focus for this study. Readers already familiar with the Geographic Data Language should note here that British Rail's naming conventions have been abandoned in order to simplify the presentation of the main ideas disussed in this article.

2.1 The Internal State

The internal state of the SSI is a collection of control variables (bits) whose status, it is assumed, always reflects the current state of the network. Here *routes* are fixed paths in the network (e.g. R_{53}), while *sub-routes* are components of routes that traverse a single track section (e.g. X_{ac}). *Track circuits* indicate whether or not a train is standing in the corresponding track section, while *points* record the setting of point switches (like P_2 in section X.) A route will be *set* whenever the signalman has allocated that route for a particular train; the component sub-routes will be set if the route is set. Sub-routes are *cleared* as the train proceeds along the allocated route; track circuits are set or cleared as the train enters or leaves the track section to which they refer. Points remain in one of two states: *normal* or *reverse.* R_{53} requires the points normal.

Since these control bits, or registers of the SSI are used in a similar way to program variables the standard CCS scheme for representing a datum D is chosen:

$$\mathsf{Loc}_D \stackrel{def}{=} \mathsf{put}_D(y).\mathsf{Reg}_D(y) \tag{1}$$

$$\mathsf{Reg}_D(y) \stackrel{def}{=} \mathsf{put}_D(x).\mathsf{Reg}_D(x) + \overline{\mathsf{get}}_D(y).\mathsf{Reg}_D(y) \tag{2}$$

D ranges over all track circuits, routes, sub-routes and points. In the interest of clarity the agent $\mathsf{Reg}_{X_{ac}}$ (say) will, in the following account, simply be referred to as X_{ac}. The internal state of the SSI is a large parallel composition of these agents:

$$\mathsf{InternalState} \stackrel{def}{=} (\mathsf{X}_{ac} \,|\, \mathsf{Y}_{ab} \,|\, \ldots \mathsf{R}_{53} \,|\, \mathsf{R}_{51} \,|\, \ldots \mathsf{W} \,|\, \mathsf{X} \,|\, \ldots \mathsf{P}_1 \,|\, \mathsf{P}_2 \,|\, \ldots) \tag{3}$$

Note that there is no internal communication between any of these components.

2.2 Operational Aspects of the SSI

To permit the passage of trains the signalman may *request* a route and the Interlocking will determine if it is safe to set up that route by consulting the internal state of the SSI. Setting up the route involves a change in the internal state (i.e. setting some control bits) and the sending of a command telegram to the appropriate track-side modules to move points switches to the required positions. This change of state in the signalling system is governed by rules in the Geographic Data as described in Section 3.

Another activity initiated by the signalman is that of route *cancellation.* In normal operating circumstances a number of preconditions must be met before a route may be cancelled. For this study, however, it is assumed that *routes may be cancelled unconditionally and at any time.* Clearing the appropriate control bit (e.g. R_{53}) has the knock-on effect of clearing all sub-routes on the route, down to the first occupied track section. The third function of interest is that in which the SSI acts as a *monitor* – although the only concern here is that the SSI monitors the positions of trains. Clearing a track circuit may also have the knock-on effect of clearing appropriate sub-routes.

The overall input/output behaviour of the SSI is summarised in Figure 2. However, in this article signals are treated abstractly – they simply mark the start and the end routes. This arises because trains are not assumed to obey signals. In fact *track circuits may change state unconditionally and at any time.*

The $\mathsf{Control}$ agent is defined as the sum of the three behaviours mentioned above:

$$\mathsf{Control} \stackrel{def}{=} \mathsf{RouteRequest} + \mathsf{RouteCancel} + \mathsf{Monitor} \tag{4}$$

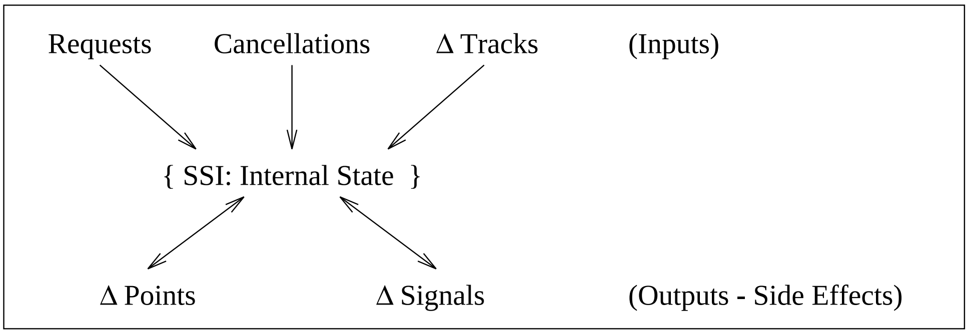

Figure 2: *Input-Output behaviour of the SSI*

where each component encodes appropriate Geographic Data as described below. The $\mathsf{Control}$ and the $\mathsf{InternalState}$ are then composed as follows:

$$\mathsf{Interlocking} \stackrel{def}{=} (\mathsf{Control} \,|\, \mathsf{InternalState}) \setminus \mathsf{L} \tag{5}$$

The visible actions represent the buttons on the signalman's control panel and certain status indicators – all connections to the network have been abstracted.

3 Geographic Data

The rules expressed in the Geographic Data Language are essentially (static) conditional statements where the current state of the SSI dictates whether certain actions can be performed. The following narrative is focused on the network subcomponent shown in Figure 1, where R_{53} proceeds over sub-routes X_{ab} and Z_{ab}.

3.1 Setting Routes

The data used in setting up a route are 'route request' and 'points free' data.

Route Request Data: Reproduced here is the route request rule for R_{53}:

Q_{53}	*if*	P_2 *cnf*,	Z_{ba} 0,	X_{ba} 0	
	then	P_2 *cn*,	X_{ab} 1,	Z_{ab} 1,	R_{53} 1

The meaning of this is as follows: if the *tests* (the precondition) all succeed then perform the *commands* (the postcondition) to set up the route.

Of the three tests in this rule the first is a points check – are the points *controlled normal* or are they *free* to go normal? That is:

$$P_2\ cnf \iff P_2\ cn\ \vee\ P_2(N),$$

What it means for a point to be free is indicated by the 'points free' data described below. The other two tests in Q_{53} check the status of sub-routes which oppose R_{53}. The careful reader will note that the test '$P_2\ cn$' seems redundant since, intuitively, if the points *are* normal they should be *free to go* normal. That is to say $P_2\ cn \implies P_2(N)$. This idea is explored further in Section 4 where some of the properties the Data should enjoy are presented.

Points Free Data: These data indicate when it is permissible to switch points from one controlled position to another. For example:

$P_2(N)$	*if* $X_{ac}\ 0,\quad X_{ca}\ 0$
$P_2(R)$	*if* $X_{ab}\ 0,\quad X_{ba}\ 0$

$P_2(N)$ states that the points can be moved to the normal position if neither of the sub-routes over the point in the reverse sense are set (see Figure 1.)

It is helpful to adopt the view that a sub-route, when set over a section containing points, *controls* the points in one position or the other. Thus, when '$X_{ac}\ 1$' the points are controlled reverse; if neither '$X_{ac}\ 1$' nor '$X_{ca}\ 1$' the points are free to go normal.

In filling out the definition of RouteRequest, in Equation (4), the route request rules are translated more or less directly into CCS as follows:

$$\begin{aligned}
\mathsf{RouteRequest} &\stackrel{def}{=} \mathsf{set}(r).\mathsf{Test}(r) && (6)\\
\mathsf{Test}(r) &\stackrel{def}{=} \mathsf{if}\ (r = R53)\ \mathsf{then}\\
&\quad\ \mathsf{get}_{X_{ba}}(u).\mathsf{get}_{Z_{ba}}(v).\mathsf{if}\ (u = 0 \wedge v = 0)\ \mathsf{then}\ \ \mathsf{P2N}(r)\ \mathsf{else}\ \ \mathsf{Fail}\\
&\ \ + \ \mathsf{if}\ (r = R51)\ \mathsf{then}\\
&\quad\ \mathsf{get}_{X_{ca}}(u).\mathsf{get}_{W_{ba}}(v).\mathsf{if}\ (u = 0 \wedge v = 0)\ \mathsf{then}\ \ \mathsf{P2R}(r)\ \mathsf{else}\ \ \mathsf{Fail}\\
&\ \ + \ \mathsf{if}\ \cdots && (7)
\end{aligned}$$

In the definition of the $\mathsf{Test}(r)$ the parameter is matched in one of the summand's branches only. The first two control variables accessed correspond to the tests on opposing sub-routes mentioned above; if these succeed the points test is invoked:

$$\begin{aligned}
\mathsf{P2N}(r) &\stackrel{def}{=} \mathsf{get}_{P_2}(u).\mathsf{if}\ (u = cn)\ \mathsf{then}\ \ \mathsf{SetUp}(r)\ \mathsf{else}\ \ \mathsf{get}_{X_{ac}}(v).\mathsf{get}_{X_{ca}}(w).\\
&\quad\ \mathsf{if}\ (v = 0 \wedge w = 0)\ \mathsf{then}\ \ \mathsf{SetUp}(r)\ \mathsf{else}\ \ \mathsf{Fail} && (8)\\
\mathsf{P2R}(r) &\stackrel{def}{=} \mathsf{get}_{P_2}(u).\mathsf{if}\ (u = cr)\ \mathsf{then}\ \ \mathsf{SetUp}(r)\ \mathsf{else}\ \ \mathsf{get}_{X_{ab}}(w).\mathsf{get}_{X_{ba}}(w).\\
&\quad\ \mathsf{if}\ (v = 0 \wedge w = 0)\ \mathsf{then}\ \ \mathsf{SetUp}(r)\ \mathsf{else}\ \ \mathsf{Fail} && (9)
\end{aligned}$$

Notice how this agent is translated from the 'points free' data above. The rationale governing the order in which the points check is carried out – i.e. test the points bit first and if this fails access the 'points free' data – simply follows from the order in which the SSI's control program performs the tests.

State $\mathsf{SetUp}(r)$ is reached if all tests in the rule succeed; the Fail state is reached otherwise, and in this case the overall state of the signalling system remains unchanged:

$$\mathsf{SetUp}(r) \stackrel{def}{=} \mathsf{if}\ (r = R53)\ \mathsf{then}\ \ \overline{\mathsf{put}}_{P_2}(cn).\overline{\mathsf{put}}_{X_{ab}}(1).\overline{\mathsf{put}}_{Z_{ab}}(1).\overline{\mathsf{put}}_{R_{53}}(1).\mathsf{Success}$$

$$+ \quad \mathsf{if} \cdots \tag{10}$$

$$\mathsf{Success} \stackrel{def}{=} \overline{\mathsf{success}}.\mathsf{Control} \tag{11}$$

$$\mathsf{Fail} \stackrel{def}{=} \overline{\mathsf{fail}}.\mathsf{Control} \tag{12}$$

Again, in $\mathsf{SetUp}(r)$ the parameter is matched in one branch only – the InternalState then being updated appropriately.

3.2 Cancelling Routes

The definition of RouteCancel is straightforward enough since routes may be cancelled unconditionally. The signalman's involvement need not be represented:

$$\mathsf{RouteCancel} \stackrel{def}{=} \overline{\mathsf{put}}_{R_{53}}(0).\mathsf{Control} + \overline{\mathsf{put}}_{R_{51}}(0).\mathsf{Control} + \cdots + \mathsf{Unlock} \tag{13}$$

Once a route has been cancelled, the first sub-route may be cleared, then the second, then the third, and so on. This mechanism is described by the sub-route release data:

Sub-route Data: Sub-routes are released in the order they are traversed:

X_{ca} 0	*if* X 0, R_{51} 0
Y_{ba} 0	*if* Y 0, X_{ca} 0, X_{ba} 0

There are two cases here. The first sub-route on a route is cleared once the route and the first track circuit have been cleared; other sub-routes are cleared if the track circuit and all preceding sub-routes are cleared.

The sub-route release mechanism is captured in the agent Unlock which has one branch corresponding to each sub-route release rule specified:

$$\begin{aligned}
\mathsf{Unlock} \stackrel{def}{=}\ & \mathsf{get}_X(u).\mathsf{get}_{R_{53}}(v). \\
& \mathsf{if}\ (u = 0 \wedge v = 0)\ \mathsf{then}\ \overline{\mathsf{put}}_{X_{ab}}(0).\mathsf{Control}\ \mathsf{else}\ \mathsf{Control} \\
+\ & \mathsf{get}_Z(u).\mathsf{get}_{X_{ab}}(v). \\
& \mathsf{if}\ (u = 0 \wedge v = 0)\ \mathsf{then}\ \overline{\mathsf{put}}_{Z_{ab}}(0).\mathsf{Control}\ \mathsf{else}\ \mathsf{Control} \\
+\ & \mathsf{get}_W(u).\mathsf{get}_{X_{ca}}(v).\mathsf{get}_{X_{ba}}(w). \\
& \mathsf{if}\ (u = 0 \wedge v = 0 \wedge w = 0)\ \mathsf{then}\ \overline{\mathsf{put}}_{W_{ba}}(0).\mathsf{Control}\ \mathsf{else}\ \mathsf{Control} \\
+\ & \cdots
\end{aligned} \tag{14}$$

This widely branching agent may seem a little unnatural but the reader is reminded that the aim is to express to Interlocking's logic – the next possible actions of the control – rather than to model the behaviour of the SSI explicitly. It turns out that the implementation of this control system performs a check on each data block associated with sub-route release during each polling cycle. The definition above captures this possibility without specifying how or when these data are accessed.

3.3 Network Monitoring

The third component of the $\mathsf{Control}$ is the $\mathsf{Monitor}$ which updates the internal state whenever a track circuit changes state. Define:

$$\begin{aligned} \mathsf{Monitor} \stackrel{def}{=} & \ \overline{\mathsf{put}}_Y(1).\mathsf{Control} + \overline{\mathsf{put}}_X(1).\mathsf{Control} + \cdots \\ + & \ \overline{\mathsf{put}}_Y(0).\mathsf{Control} + \overline{\mathsf{put}}_X(0).\mathsf{Control} + \cdots \end{aligned} \tag{15}$$

This monitoring agent is quite abstract since it has no explicit connection to the network. Rather, it encodes the idea that track circuits can change state at random.

Clearly the assumption that track circuits may change state unconditionally is quite outrageous since no real network can have this degree of freedom. However, if it is possible to demonstrate that this chaotic system has certain desirable safety properties then the results will carry over into the more restricted domain. The benefit lies in being able to show something of the robustness of the SSI's internal representation of the network it controls. At least the safety features of the SSI examined here will have been demonstrated without recourse to any assumptions about the expected, orderly behaviour of the network.

Before concluding this section note that in this CCS encoding of the Interlocking there is an implicit assumption about the granularity of the model since it is only in the $\mathsf{Control}$ states that the system can respond to external stimuli. These are the observable states of the SSI which should satisfy the safety criteria discussed below.

4 Safety Properties of the Geographic Data

The primary concern is to ensure that trains which correctly obey signals do not find themselves in dangerous situations. Trains may find themselves on a collision course if two or more routes are set simultaneously across a track section; trains may be derailed by points switches if the points are incorrectly aligned for a particular routing. Three properties of interest here are as follows:

- P: No more than one of the sub-routes over a track section is set at any one time.
- Q: If a route is set, then all its component sub-routes are set.
- R: If a sub-route over a track section containing points is set, then the points are correctly aligned with that sub-route.
- R': A point controlled reverse (resp. normal), is free to go reverse (normal.)

If the Interlocking does not have these properties the implication is that there is an error in the SSI's geographic database. R is a little difficult to understand perhaps but a glance at Figure 1 should convince the reader it has the following interpretation: if either X_{ac} or X_{ca} are set then P_2 should be reverse; if either X_{ab} or X_{ba} are set then P_2 should be normal. It turns out that if we take points to be boolean valued variables then R is equivalent to R'.

4.1 Analysis of Invariants

The safety properties of a system can often be expressed as invariants. These are normally represented by equations relating the state variables of the system, equations which should remain true over all execution sequences. The analysis here concentrates on the Control agent, which has only a few hundred states, and disregards the InternalState altogether. We suppose that the purely CCS description of the SSI is augmented with a collection of (zero-one) variables to *represent* the internal state. The properties of interest may then be informally expressed, using an obvious notation, as follows:

Encoding P**:** Since the sub-routes over a particular track section (e.g. X) are variables whose values are either zero or one, the required property for X is:

$$X_{ab} + X_{ac} + X_{ba} + X_{ca} \dot{-} 1 = 0.$$

Similarly for the other track sections. This may be represented more schematically by $\sum_i X_i \dot{-} 1 = 0$, as long as the range of the subscript is understood. Then P is captured by:

$$I_P \iff \bigwedge_X \left(\sum_i X_i \dot{-} 1 = 0 \right)$$

Encoding Q**:** This is more difficult as it is necessary to know how routes are defined in terms of their sub-routes. The route request data encodes this information of course, but an independent source is needed to write down the predicate:

$$\begin{aligned} I_Q \iff & (R_{53} = 1 \Rightarrow X_{ab} = 1 \wedge Z_{ab} = 1) \wedge \\ & (R_{26} = 1 \Rightarrow X_{ca} = 1 \wedge Y_{ba} = 1 \wedge U_{ba} = 1) \wedge \cdots \end{aligned}$$

Encoding R**:** A similar difficulty arises here also since it is necessary to know which sub-routes control the points in which positions. Assuming this to be independently specified, define:

$$\begin{aligned} I_R \iff & (X_{ab} = 1 \vee X_{ba} = 1 \Rightarrow P_2 = cn) \wedge \\ & (X_{ac} = 1 \vee X_{ca} = 1 \Rightarrow P_2 = cr) \wedge \cdots \end{aligned}$$

Note that P_2 is also a binary variable but in the interests of clarity its values are referred to as cn and cr as usual.

The invariant, then, is the conjunct $I \iff I_P \wedge I_Q \wedge I_R$.

4.2 Formalising Invariants in L_μ

The Interlocking has been constructed so that it may be submitted to a tool like the Concurrency Workbench [CPS89] for automatic analysis. If we can capture the desired properties of the system in a formula of the modal mu-calculus, our favourite model checker will verify whether on not the $\mathsf{Interlocking}$ satisfies the formula. That is to say:

$$(\mathsf{Control} \,|\, \mathsf{InternalState}) \setminus \mathsf{L} \quad \models \quad \nu Z.F \wedge [\_]Z \qquad (*)$$

where F formally encodes the desired properties of the system which we expressed above in the formula I. It turns out that a technical difficulty arises in formalising these *state* properties (P, Q, and R above) for the model constructed in Section 2 since it does not permit the direct observation of the internal state of the machine. The restriction ($\setminus \mathsf{L}$) in Eqn. 5 effectively hides the $\mathsf{InternalState}$ from view. It is, however, less than desirable to leave the get_D and $\overline{\mathsf{put}}_D$ actions visible since no external agent should be permitted to interact with the SSI in this fashion.

The simplest method of addressing this difficulty is to redefine the register agents from which the $\mathsf{InternalState}$ is composed so that they may, at any time, report their contents:

$$\mathsf{Reg}_D(y) \stackrel{def}{=} \mathsf{put}_D(x).\mathsf{Reg}_D(x) + \overline{\mathsf{get}}_D(y).\mathsf{Reg}_D(y) + \overline{\mathsf{obs}}_D(y).\mathsf{Reg}_D(y)$$

The $\overline{\mathsf{obs}}_D$ actions do not change the state of the machine in any way – they can therefore be used to interrogate the internal state of the SSI. No agent need be defined to perform this interrogation however, since we merely wish to exploit the idea than in any state it is now *possible* for a datum D to report its value. The safety properties described above may now be formalised along the following lines:

Formalising P: The idea that the sub-routes over the track section Z (say) are never set together can be expressed by the modal formula:

$$\langle \overline{\mathsf{obs}}_{Z_{ab}}(0) \rangle \mathtt{tt} \;\vee\; \langle \overline{\mathsf{obs}}_{Z_{ba}}(0) \rangle \mathtt{tt}$$

which is only false if both variable hold the value 1. (The language can not quantify over actions and agent as suggested by the informal expression of this invariant.)

Formalising Q: There are several possible formulations for conditional properties such as this. We can use the derived form:

$$\langle \overline{\mathsf{obs}}_{R_{53}}(1) \rangle \mathtt{tt} \;\Rightarrow\; (\langle \overline{\mathsf{obs}}_{X_{ab}}(1) \rangle \mathtt{tt} \;\wedge\; \langle \overline{\mathsf{obs}}_{Z_{ab}}(1) \rangle \mathtt{tt})$$

but the conjunctive normal form will probably be more suitable if we are to automate the proof of this invariant:

$$([\overline{\mathsf{obs}}_{R_{53}}(1)]\mathtt{ff} \;\vee\; \langle \overline{\mathsf{obs}}_{X_{ab}}(1) \rangle \mathtt{tt}) \;\wedge\; ([\overline{\mathsf{obs}}_{R_{53}}(1)]\mathtt{ff} \;\vee\; \langle \overline{\mathsf{obs}}_{Z_{ab}}(1) \rangle \mathtt{tt}).$$

Formalising R: Similarly for the points property, the conjunctive normal form will likely be most useful:

$$([\overline{\mathsf{obs}}_{X_{ab}}(1)]\mathtt{ff} \;\vee\; \langle \overline{\mathsf{obs}}_{P_2}(cn) \rangle \mathtt{tt}) \;\wedge\; ([\overline{\mathsf{obs}}_{X_{ba}}(1)]\mathtt{ff} \;\vee\; \langle \overline{\mathsf{obs}}_{P_2}(cn) \rangle \mathtt{tt}).$$

The (modal) formula F, then, is just the conjunct of such 'local' invariants.

4.3 Automated Verification of Safety Properties

Not too suprisingly, in trying to verify the relation ($*$) we run into a state space explosion. While the CCS system is certainly finite the InternalState alone has 2^N possible configurations – since there are N components. This outrageous complexity effectively puts automatic analysis of Interlocking beyond the reach of finite state (global) model checkers which must represent these states explicitly. Unfortunately, the model constructed here (Eqn. 5) appears to be pathalogical in the sense that the *local* model checker of [SW89] is also forced to examine the global state space of the system.

Clearly, 2^N does not represent the inherent complexity of the data correctness problem as the vast majority of these states are unreachable. The difficulty lies in identifying the actual behaviour of the system – the intrinsic state space of the SSI – among this chaos. The key is to note that the Control defines exactly those states which are reachable from any given initial state: if the initial state is valid with respect to the safety criteria mentioned then it is enough to show that each reachable state is valid. There are many ways of capitalising on this feature of the data correctness problem, both mechanically or otherwise. The approach here focuses on the analysis of the invariants of Section 4.1 above and, although the proof described below is constructed by hand, a similar, tableaux method is surely mechanisable.

5 Proving the Invariant

In order to proceed with the proof of the invariant defined above, the state variables are gathered together in the vector

$$S = < X_{ab}, X_{ac}, \ldots \; R_{53}, R_{51}, \ldots \; X, Y, \ldots \; P_1, P_2 >$$

which is attached to the Control agent as a parameter. The goal is then to show that Control(S_0) satisfies the invariant I of Section 4.1, where S_0 is the initial state of the SSI. This is achieved indirectly (i.e. without resorting to using a model checker) as follows: first show that the initial state satisfies the I. Next, since the Control( )is deterministic, examine all traces from a state Control(S) to its successor states, Control(S'). The obligation is to show that if Control(S) $\models I$ then Control(S') $\models I$. In the following the notation $S \models I$ is used as an abbreviation for this.

5.1 The Initial State

The initial state of the system is that in which all routes and sub-routes are cleared (zero), all track circuits are clear and all points are normal:

$$S_0 = < 0, 0, 0 \ldots \; cn, cn > .$$

That $S_0 \models I_P$ is easy to see since $\forall X. \sum_i X_i = 0$. It is also trivial that $S_0 \models I_Q$ since each conjunct reduces to *false* $\Rightarrow$ *false*, which is *true*. Similarly $S_0 \models I_R$, so the conclusion that $S_0 \models I_P \wedge I_Q \wedge I_R$ follows immediately.

5.2 Sub-Route Release Rules

In the agent Unlock, Equation (14), each branch encodes one of the sub-route release rules and the only update to the internal state occurs through an action of the form $\overline{\mathsf{put}}_D(0)$. From the definition of I_P it can be seen that *clearing* sub-routes does not affect the truth of I_P. As the same is true of I_R it only remains to show that I_Q is invariant. Also, since routes may be cancelled unconditionally there is no work to do in showing that the invariant is preserved in, say, updating S to $S[0/R_{53}]$. Clearly, if $S \models R_{53} = 1 \Rightarrow X_{ab} = 1 \wedge Z_{ab} = 1$ then $S[0/R_{53}]$ does so vacuously.

To see how to check the sub-route release rules, consider the following examples:

$$\begin{aligned}
\mathsf{Control}(S) &= \mathsf{get}_X(0).(\mathsf{get}_{R_{53}}(0).\overline{\mathsf{put}}_{X_{ab}}(0).\mathsf{Control}(S') + \mathsf{get}_{R_{53}}(1).\mathsf{Control}(S)) \\
&+ \mathsf{get}_X(1)\ldots && \text{(A)} \\
\mathsf{Control}(S) &= \mathsf{get}_Z(0).(\mathsf{get}_{X_{ab}}(0).\overline{\mathsf{put}}_{Z_{ab}}(0).\mathsf{Control}(S'') + \mathsf{get}_{X_{ab}}(1).\mathsf{Control}(S)) \\
&+ \mathsf{get}_Z(1)\ldots && \text{(B)}
\end{aligned}$$

The components left undefined in (A) and (B) evolve back to the state $\mathsf{Control}(S)$. Now, where $S' = S[0/X_{ab}]$ and $S'' = S[0/Z_{ab}]$, the obligation is to show that both S' and S'' satisfy the I as long as $S \models I$.

To see that $S' \models I_Q$ note that only one conjunct of I_Q is affected in updating S to S', namely $R_{53} = 1 \Rightarrow X_{ab} = 1 \wedge Z_{ab} = 1$. Then

$$\begin{aligned}
&S \models R_{53} = 1 \Rightarrow X_{ab} = 1 \wedge Z_{ab} = 1 && \{\text{since } S \models I_Q\} \\
&S' \models R_{53} = 0 \wedge X_{ab} = 0 && \{\text{from (A)}\}
\end{aligned}$$

The conclusion that $S' \models I_Q$ follows immediately since, if $R_{53} = 0$, the above implication is always true. The same conjunct is involved in updating S to S'', but now there are two cases to consider which depend on whether or not R_{53} is set in S. Firstly:

$$\begin{aligned}
&S \models R_{53} = 1 && \{\text{hypothesis}\} \\
&S \models R_{53} = 1 \Rightarrow X_{ab} = 1 \wedge Z_{ab} = 1 && \{\text{since } S \models I_Q\} \\
&S \models X_{ab} = 1 \wedge Z_{ab} = 1 && \{\text{conjoining the above}\}
\end{aligned}$$

but from (B) it is clear that $S \models X_{ab} = 0$, which contradicts $S \models R_{53} = 1$. Proceeding to the second case:

$$\begin{aligned}
&S \models R_{53} = 0 && \{\text{hypothesis}\} \\
&S'' \models R_{53} = 0 \wedge X_{ab} = 0 \wedge Z_{ab} = 0 && \{\text{from (B)}\}
\end{aligned}$$

However, the conclusion that $S'' \models R_{53} = 1 \Rightarrow X_{ab} = 1 \wedge Z_{ab} = 1$ now follows immediately since $S'' \models R_{53} = 0$. Hence, $S'' \models I_Q$.

To see how errors might be identified suppose that in (B), for example, the variable X_{ba} was spuriously tested instead of X_{ab}. Case analysis would fail to find the contradiction when $R_{53} = 1$, so:

$$S \models R_{53} = 1 \wedge X_{ab} = 1 \wedge Z_{ab} = 1 \qquad \{\text{as before}\}$$

but clearly $S'' \not\models R_{53} = 1 \Rightarrow X_{ab} = 1 \wedge Z_{ab} = 1$, since $S'' \models Z_{ab} = 0$. That $X_{ba} = 0$ is irrelevant. Errors in the conclusions of the rules are identified in a similar way.

The sub-route release rules encode one view of the connectivity of the network described in the Geographic Data; the route request rules provide another. Having verified that the sub-route release rules are valid, appeal can be made to a series of lemmas like:

$$X_{ab} = 1 \Rightarrow Z_{ab} = 1 \quad \& \quad X_{ba} = 1 \vee X_{ca} = 1 \Rightarrow Y_{ba} = 1 \quad \& \quad Y_{ba} = 1 \Rightarrow U_{ba} = 1 \quad \&\text{c.}$$

These will be of use in demonstrating that the route request rules are valid.

5.3 Route Request Rules

Consider the following derivation from the state $S \models I$:

$$\begin{array}{rcll}
\mathsf{Control}(S) & = & \mathsf{set}(R53).\mathsf{Control}^1_{R53}(S) & \\
\mathsf{Control}^1_{R53}(S) & = & \mathsf{get}_{X_{ba}}(0).(\mathsf{get}_{Z_{ba}}(0).\mathsf{Control}^2_{R53}(S) + \mathsf{get}_{Z_{ba}}(1).\mathsf{Fail}(S)) & \\
 & + & \mathsf{get}_{X_{ba}}(1)\ldots & \text{(C)} \\
\mathsf{Control}^2_{R53}(S) & = & \mathsf{get}_{P_2}(cr).\mathsf{Control}^3_{R53}(S) + \mathsf{get}_{P_2}(cn).\mathsf{Control}^4_{R53}(S) & \text{(D)} \\
\mathsf{Control}^3_{R53}(S) & = & \mathsf{get}_{X_{ac}}(0).(\mathsf{get}_{X_{ca}}(0).\mathsf{Control}^4_{R53}(S) + \cdots) + \cdots & \text{(E)} \\
\mathsf{Control}^4_{R53}(S) & = & \overline{\mathsf{put}}_{P_2}(cn).\overline{\mathsf{put}}_{X_{ab}}(1).\overline{\mathsf{put}}_{Z_{ab}}(1).\overline{\mathsf{put}}_{R_{53}}(1).\mathsf{Success}(S') & \\
 & & & \\
\mathsf{Fail}(S) & = & \overline{\mathsf{fail}}.\mathsf{Control}(S) & \\
\mathsf{Success}(S) & = & \overline{\mathsf{success}}.\mathsf{Control}(S) &
\end{array}$$

where $S' = S[cn/P_2,\ 1/X_{ab},\ 1/Z_{ab},\ 1/R_{53}]$. The reader can check these derivations against Equations (6) to (12), but note that the summands left undefined in (C) and (E) all evolve to $\mathsf{Fail}(S)$. Given that $S \models I_P \wedge I_Q \wedge I_R$:

$$\begin{array}{lr}
S \models Z_{ab} + Z_{ba} \dot{-} 1 = 0 & \{\text{since } S \models I_P\} \\
S \models Z_{ab} \dot{-} 1 = 0 & \{\text{from (C) } S \models Z_{ba} = 0\}
\end{array}$$

The only sub-route over Z which is set in updating S to S' is Z_{ab} so, as $1 \dot{-} 1 = 0$, it follows that $S' \models \sum_i Z_i \dot{-} 1 = 0$. Further:

$$\begin{array}{lr}
S \models X_{ab} + X_{ac} + X_{ba} + X_{ca} \dot{-} 1 = 0 & \{\text{since } S \models I_P\} \\
S \models X_{ab} + X_{ac} + X_{ca} \dot{-} 1 = 0 & \{\text{from (C)}\}
\end{array}$$

From (D) there are two cases to consider for either $S \models P_2 = cn$ or $S \models P_2 = cr$:

$$\begin{array}{lr}
S \models P_2 = cn & \{\text{by hypothesis}\} \\
S \models P_2 = cn \Rightarrow X_{ac} = 0 \wedge X_{ca} = 0 & \{\text{since } S \models I_R \text{ and } R \iff R'\} \\
S \models X_{ab} \dot{-} 1 = 0 &
\end{array}$$

The only sub-route over X which is set in updating S to S' is X_{ab} so, as $1 \dot{-} 1 = 0$, it follows that $S' \models \sum_i X_i \dot{-} 1 = 0$. For the other case:

$$\begin{array}{lr}
S \models P_2 = cr & \{\text{by hypothesis}\} \\
S \models P_2 = cr \Rightarrow X_{ab} = 0 \wedge X_{ba} = 0 & \{\text{since } S \models I_R\} \\
S \models X_{ac} + X_{ca} \dot{-} 1 = 0 & \\
S \models \sum_i X_i = 0 & \{\text{since from (E) } S \models X_{ac} + X_{ca} = 0\}
\end{array}$$

Again, the only sub-route over X which is set in updating S to S' is X_{ab} so, as $0 \dot{-} 1 = 0$, it follows that $S' \models \sum_i X_i \dot{-} 1 = 0$.

Since $S' \models \sum_i Z_i \dot{-} 1 = 0 \wedge \sum_i X_i \dot{-} 1 = 0$ and only these two components of I_P are affected, the conclusion that $S' \models I_P$ follows.

That $S' \models I_Q$ is easy to see. In this example there is only one conjunct which is affected, namely $R_{53} = 1 \Rightarrow X_{ab} = 1 \wedge Z_{ab} = 1$, and clearly:

$$S[cn/P_2, 1/X_{ab}, 1/Z_{ab}, 1/R_{53}] \models R_{53} = 1 \Rightarrow X_{ab} = 1 \wedge Z_{ab} = 1$$

It is also easy to see $S' \models I_R$ since only P_2 is updated:

$$S \models X_{ac} = 0 \wedge X_{ca} = 0 \qquad \{\text{from (C), (D) and (E)}\}$$
$$S \models (P_2 = cn \Rightarrow X_{ac} = 0 \wedge X_{ca} = 0) \wedge (P_2 = cr \Rightarrow X_{ab} = 0 \wedge X_{ba} = 0) \quad \{S \models I_R\}$$

Now S' satisfies the right-hand conjunct since $S' \models P_2 = cn \wedge X_{ab} = 1$; the left-hand conjunct is satisfied since only X_{ab} and Z_{ab} are set in updating S to S' and so the truth of $X_{ac} = 0 \wedge X_{ca} = 0$ is unaffected. Hence $S' \models I$ if $S \models I$.

To see how errors in the data may be revealed, consider the step in the proof above which showed $Z_{ab} + Z_{ba} \dot{-} 1 = 0$ to be invariant. Now, when $S' = S[1/Z_{ab}, \ldots]$, the following question arises: is there an assignment of the variables (in S) such that, on substituting the known values (in S'), the formula is falsifiable? It is necessary to ascertain whether or not the conjecture $S' \models Z_{ba} = 0$ is refutable.

In this case the contextual information in (C) asserts that $S \models Z_{ba} = 0$, so the conjecture is irrefutable. But suppose that in (C), for example, it is found that $S \models W_{ba} = 0$. Case analysis on Z_{ba} quickly reveals that $S' \models Z_{ba} = 0$ is unmet in the case $S \models Z_{ba} = 1$. Note that it is only necessary to consider variables *local* to that which is updated – i.e. refer only to sub-formulae of I in which the updated variable occurs.

5.4 Programming Pragmatics

Tautology checking in general is NP-complete. Here, to overcome the exponential character of the satisfaction problem, *contextual* information is obtained in deriving Control(S') from Control(S). This context instantiates certain, key variables, so limiting the space to be searched in order to establish each step in proving the invariant.

The highly regular structure of the Control is reflected in regularity in the structure of the proof sketched above. This is of practical interest since each step in the proof corresponds to verifying one of the rules in the Geographic Data. The benefit here is in being able to pinpoint exactly where errors occur, but the major advantages will be seen on considering how to mechanise the proof process. In the end, a proof tool is sought which will assist signalling engineers in verifying their designs and which can be used easily and efficiently to analyse data-sets of varying complexity.

Pragmatic issues such as finding a fast algorithm for checking the Geographic Data certainly lie at the heart of the problem. Indeed, the search for a reasonable formalism for representing the problem has been lead throughout by the desire to mechanise the proof procedure. Clearly, any efforts to automate the 'invariants' proof will have been wasted if it remains impossible to analyse large data-sets. This leads to a question of the composition, or decomposition of railway signalling systems, which we now briefly explore.

6 Composition & Decomposition

To all intents and purposes the foregoing discussion has concentrated on the analysis of an isolated, and rather small, railway signalling system. To break off the story at this point would be somewhat misleading as there are important issues which cannot be addressed by looking at an isolated system. Primarily there is the idea that the control of a complex signalling area will be distributed between several SSI's, and these must act in consort to ensure the safety of rail traffic (see Cribbens [Cri87].) Although the informal specification [Bri88] is a little vague about the behaviour anticipated at the edges of the region controlled by a single SSI, it is at least clear that routes will have to be set, and cleared, across these boundaries. On the other hand, Cribbens suggests that in a typical installation a single SSI would control an area of about 50 track sections (or 70 routes) over some 15 kilometres of track. Given that a mechanical approach to the Geographic Data Correctness problem is sought, the reader will appreciate that it will still be necessary to decompose these data-sets in some way, in order to make the analysis tractable at all.

6.1 Agent Decomposition

Intuitively, the effects of a change in some part of the signalling system will be entirely local – setting a particular route will only effect the availability of a small number of adjacent (or incident) routes. This leads us to believe that decomposition of the verification task into *local* invariant problems will be a fruitful strategy to follow in solving the *global* problem. Here, we sketch one approach to decomposition that is suggested by the route structure of the geographic data. In order to make these ideas a little clearer, attention is drawn to the layout depicted in Figure 3 which is schematically divided into two parts.

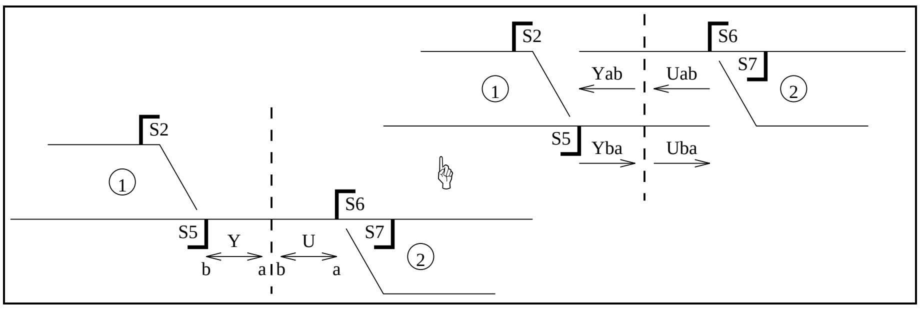

Figure 3: *Composing or Decomposing data-sets: Simple partitioning.*

The relevant Geographic Data are partitioned as indicated in the figure such that the left-hand system is responsible for R_{26} – setting the route, and releasing its component sub-routes – while the right-hand system is responsible for R_{75}. The bi-directional track (Y–U) is under the control of both systems. Note that the left-hand system must access (read) certain components of the right-hand system's internal state, and vice versa. In detail: $\mathsf{Control}_1$ must read from $\mathsf{InternalState}_2$ the components Y_{ab} and U in order to set R_{26} and clear U_{ba} respectively; meanwhile, $\mathsf{Control}_2$ must read from $\mathsf{InternalState}_1$ the components U_{ba} and Y in order to set R_{75} and clear Y_{ab}. The $\mathsf{InternalState}$ agents of the two systems are disjoint:

$$\begin{aligned}\mathsf{InternalState}_1 &\stackrel{def}{=} (\,\mathsf{Y}_{ba}\,|\,\mathsf{U}_{ba}\,|\,\ldots \mathsf{R}_{53}\,|\,\mathsf{R}_{26}\,|\,\ldots \mathsf{Y}\,|\,\mathsf{X}\,|\,\ldots \mathsf{P}_1\,|\,\mathsf{P}_2\,|\,\ldots) \\ \mathsf{InternalState}_2 &\stackrel{def}{=} (\,\mathsf{U}_{ab}\,|\,\mathsf{Y}_{ab}\,|\,\ldots \mathsf{R}_{68}\,|\,\mathsf{R}_{75}\,|\,\ldots \mathsf{U}\,|\,\mathsf{V}\,|\,\ldots \mathsf{P}_3\,|\,\mathsf{P}_4\,|\,\ldots).\end{aligned}$$

Now, if we are to assemble the $\mathsf{Control}_1$ and $\mathsf{InternalState}_1$ agents along the lines suggested by Eqn. 5, then restricting on *all* the get_D and $\overline{\mathsf{put}}_D$ actions in $\mathsf{InternalState}_1$ has the unavoidable effect of hiding from $\mathsf{Control}_2$ those data that it needs to control R_{75}. However, we may use the appropriate $\overline{\mathsf{obs}}_D$ actions to communicate these data to $\mathsf{Control}_2$ (and similarly for $\mathsf{Control}_1$) thus:

$$\begin{aligned}\mathsf{SSI}_1 &\stackrel{def}{=} (\mathsf{Control}_1[\mathsf{obs}_{Y_{ab}}/\mathsf{get}_{Y_{ab}},\, \mathsf{obs}_U/\mathsf{get}_U]\,|\,\mathsf{InternalState}_1) \setminus \mathsf{L}_1 \\ \mathsf{SSI}_2 &\stackrel{def}{=} (\mathsf{Control}_2[\mathsf{obs}_{U_{ba}}/\mathsf{get}_{U_{ba}},\, \mathsf{obs}_Y/\mathsf{get}_Y]\,|\,\mathsf{InternalState}_2) \setminus \mathsf{L}_2\end{aligned}$$

One is then tempted to compose these sub-systems as follows:

$$(\mathsf{SSI}_1\,|\,\mathsf{SSI}_2)\setminus\{\mathsf{obs}_Y,\, \mathsf{obs}_U,\, \mathsf{obs}_{Y_{ab}},\, \mathsf{obs}_{U_{ba}}\}$$

However, *this* coupled system is clearly unsafe since the two controllers may simultaneously set up routes over the bi-directional track. A simple device to overcome this problem would be a 'coupling agent' – in this case a semaphore – to ensure that panel requests for the conflicting routes are never processed simultaneously.

$$\begin{aligned}\mathsf{Sem} &\stackrel{def}{=} \overline{\mathsf{acq}}.\mathsf{Sem}' \\ \mathsf{Sem}' &\stackrel{def}{=} \mathsf{rel}.\mathsf{Sem} + \mathsf{rej}.\mathsf{Sem}'\end{aligned}$$

This semaphore is only intended for use where conflict might arise between the two $\mathsf{Control}$ agents in *setting* routes across the boundary. (There is no conflict in releasing routes.) Changes in the definitions of (6) to (12) might run as follows:

$$\begin{aligned}\mathsf{RouteRequest}_1 &\stackrel{def}{=} \mathsf{set}(r).\mathsf{Acq}_1(r) \\ \mathsf{Acq}_1(r) &\stackrel{def}{=} \mathsf{if}\ (r = R26 \vee r = R46)\ \mathsf{then}\ \ (\mathsf{acq}.\mathsf{Test}_1(r) + \overline{\mathsf{rej}}.\mathsf{Rel}_1(r)) \\ &\qquad \mathsf{else}\ \ \mathsf{Test}_1(r) \\ \mathsf{Test}_1(r) &\stackrel{def}{=} \mathsf{if}\ (r = R53)\ \mathsf{then} \\ &\qquad \mathsf{get}_{X_{ba}}(u).\mathsf{get}_{Z_{ba}}(v).\mathsf{if}\ (u = 0 \wedge v = 0)\ \mathsf{then}\ \ \mathsf{P2N}(r)\ \mathsf{else}\ \ \mathsf{Rel}_1(r) \\ &\quad + \ \mathsf{if}\ (r = R26)\ \mathsf{then}\ \ \cdots \\ \\ \mathsf{Rel}_1(r) &\stackrel{def}{=} \mathsf{if}\ (r = R26 \vee r = R46)\ \mathsf{then}\ \ \overline{\mathsf{rel}}.\mathsf{Control}_1\ \mathsf{else}\ \ \mathsf{Control}_1\end{aligned}$$

Similar definitions are made in $\mathsf{Control}_2$. We may then couple these systems together as follows, hiding the semaphore actions which the two controllers share:

$$\mathsf{Sys} \stackrel{def}{=} (\mathsf{SSI}_1 \,|\, \mathsf{Sem} \,|\, \mathsf{SSI}_2)\backslash\{\mathsf{obs}_Y, \mathsf{obs}_U, \mathsf{obs}_{Y_{ab}}, \mathsf{obs}_{U_{ba}}, \mathsf{acq}, \mathsf{rel}, \mathsf{rej}\} \qquad (\dagger)$$

6.2 Recovering Global Safety Properties

Given the above decomposition into SSI_1 and SSI_2 one can proceed with the safety analysis, as sketched in Section 5, for both systems – the idea being to take the conjunction of the two safety properties as our global invariant. However, focusing on SSI_1, we can see immediately that we cannot prove the mutual exclusion property (P) for the fringe track section Y. That is, using the formal characterisation of Section 4.2, we can not prove

$$\langle\overline{\mathsf{obs}}_{Y_{ab}}(0)\rangle\mathtt{tt} \;\vee\; \langle\overline{\mathsf{obs}}_{Y_{ba}}(0)\rangle\mathtt{tt}$$

invariant because the inward sub-route Y_{ab} is under the control of SSI_2. Nevertheless, this is certainly a property that we would want the coupled system ($\dagger$) to enjoy. Can we not recover this property from the local properties of its parts? We can prove the following invariants, for SSI_1 and SSI_2 respectively:

$$\langle\overline{\mathsf{obs}}_{Y_{ba}}(1)\rangle\mathtt{tt} \;\Rightarrow\; \langle\overline{\mathsf{obs}}_{U_{ba}}(1)\rangle\mathtt{tt}$$
$$\langle\overline{\mathsf{obs}}_{U_{ab}}(1)\rangle\mathtt{tt} \;\Rightarrow\; \langle\overline{\mathsf{obs}}_{Y_{ab}}(1)\rangle\mathtt{tt}$$

However, in *conjunction* these give us only the invariant:

$$\langle\overline{\mathsf{obs}}_{Y_{ba}}(1)\rangle\mathtt{tt} \;\wedge\; [\overline{\mathsf{obs}}_{Y_{ab}}(1)]\mathtt{ff}$$

and similarly for U. But these properties are not strong enough for us to deduce that the mutual exclusion property above holds. Nor is it immediately clear how one might repair this situation, since the missing component, viz:

$$\langle\overline{\mathsf{obs}}_{Y_{ab}}(1)\rangle\mathtt{tt} \;\wedge\; [\overline{\mathsf{obs}}_{Y_{ba}}(1)]\mathtt{ff}$$

would follow only if the following property held invariantly for SSI_1:

$$\langle\overline{\mathsf{obs}}_{U_{ba}}(1)\rangle\mathtt{tt} \;\Rightarrow\; \langle\overline{\mathsf{obs}}_{Y_{ba}}(1)\rangle\mathtt{tt}$$

But this is *not* a property of the system since the formula is false for at least one valid assignment to the variables (namely, when the sub-route U_{ba} is set, and Y_{ba} is clear.)

Our conclusion then is that we can not easily demonstrate the mutual exclusion property for these fringe track sections by considering the sub-systems in isolation. We therefore resort to directly proving that these (two) local properties hold of the coupled system above. A slightly irritating technical difficulty arises since the restriction in ($\dagger$) effectively hides the two *inward* sub-routes U_{ba} and Y_{ab}, but this can be easily resolved in practice.

7 Conclusions

The model described in Section 3 comes from a fairly direct encoding of the Geographic Data Language in CCS and, although many of the SSI's operational features have been abstracted, the representation of the interlocking is consequently rather concrete. Even so, the safety properties of the system have been nicely expressed by the invariant described in Section 4. These two points have lead to a simple, yet rigorous method for checking the integrity of the SSI's data.

While the proof technique sketched in Section 5 is linear in the number of rules in the geographic database – i.e. there is one proof obligation to discharge for each rule – the sheer size of real databases leads us to conclude that a compositional approach to the data correctness problem is essential. The ideas sketched in Section 6 are a first step in this general direction. The problem seems to be not so much in the partitioning of the data, but in drawing out global safety properties from the analysis of the parts. This composition/decomposition issue therefore remains at the focus of our ongoing research effort.

One might reasonably ask whether or not it is possible to capture the "principles of railway signalling" (see [Noc80], for example) in a formalism like CCS – with a view to making general statements about their safety properties through higher abstraction. However, signalling is very much an engineering discipline and any principles laid down will be subject to widespread variation even within a single network. Therefore, any generic approach to the data correctness problem would need a powerful mechanism for handling a great many exceptions, special cases, and so on. For this reason it is thought better to proceed on a case by case basis, and to invest some effort into mechanising the proof procedure. The success of this venture depends crucially on being able to manage the computational complexities which arise, and in being able to extend the basic method to cover new types of data and new safety criteria.

Acknowledgements

I should thank the Safety Systems Unit at British Rail Research for furnishing me with the SSI documentation upon which these researches are based. Although no formal collaboration has been entered into with British Rail, I should like to thank Ian Mitchell in particular for sharing his enthusiasm and his invaluable knowledge of railway signalling. Thanks also to Stuart Anderson and Robin Milner at Edinburgh for taking the time to listen, and to Helen Simms for proof-reading the draft.

References

[AC91] Will Atkinson and Jim Cunningham. Proving properties of a safety critical system. *Software Engineering Journal*, pages 41–50, March 1991.

[Bri88] British Railways Board. A railway signalling case study for FOREST. Internal publication by B.R. Research, Safety Systems Unit, August 1988. Issue B, ELS-DOC-4314, for the high integrity systems research project.

[Bri90] British Railways Board. SSI data preparation guide. Issue SSI8003, February 1990.

[CPS89] R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A semantics-based verification tool for finite-state systems. Technical Report ECS-LFCS-89-83, University of Edinburgh, Department of Computer Science, Kings Buildings, Edinburgh EH9 3JZ, June 1989.

[Cri87] A. H. Cribbens. Solid State Interlocking (SSI): an integrated electronic signalling system for mainline railways. *IEE Proc.*, 134(3):148–158, May 1987.

[Mil90] Robin Milner. *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, 1990.

[Mit90] I. H. Mitchell. Proposal for an SSI data checking tool. Internal publication by B.R. Research, Safety Systems Unit, June 1990.

[Mor91] M. J. Morley. Modelling British Rail's Interlocking Logic – Solid State Interlocking. Internal momorandum for MPSS, March 1991.

[Noc80] O. S. Nock. *Railway Signalling – a treatise on the recent practice of British Railways.* Adam and Charles Black, London, 1980. Prepared under the direction of a committee of the Institution of Railway Signalling Engineers under the general direction of O. S. Nock. ISBN: 0-7136-2067-6.

[SW89] Colin Stirling and David Walker. Local Model Checking in the Modal Mu-Calculus. Technical report, University of Edinburgh, May 1989.