NASA Contractor Report 198353

A Mathematical Model for Railway Control Systems

D. N. Hoover
Odyssey Research Associates, Inc., Ithaca, New York

Contract NAS1-20335

June 1996

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23681-0001

A Mathematical Model for Railway Control Systems [*][†]

D. N. Hoover
Odyssey Research Associates, Inc.
301 Dates Dr.
Ithaca NY 14850-1326
Internet: hoove@oracorp.com

June 12, 1996

Abstract

We present a general method for modeling safety aspects of railway control systems. Using our modeling method, one can progressively refine an abstract railway safety model, successively adding layers of detail about how a real system actually operates, while maintaining a safety property that refines the original abstract safety property. This method supports a top-down approach to specification of railway control systems and to proof of a variety of safety-related properties.

We demonstrate our method by proving safety of the classical block control system.

[*]This research was sponsored in part by NASA Contract NAS1-20335.

[†]Thanks to Fred Schneider, Jim Caldwell, Michael Holloway, Mike Seager, Joe Profeta, Mohsen El-Agamawi, Barry Johnson, Damir Jamsek, Doug Harper, Randy Calistri-Yeh and Dana Hartman for their advice and assistance. Special thanks to John Rushby, who has been a remarkable guide to the literature on railway safety modeling.

Contents

1 Introduction

This paper presents a mathematical model of what it is that makes a railway control system safe. By *safe*, we mean that trains never collide and never attempt to cross switches that are adversely set. By *system*, we mean that our method models how trains, switches, signals, controllers, and other devices work together to ensure safety. In this paper, we do not address the details of how individual devices work, though such things could be added in further refinements of our model. One could say that in this paper we are concerned with proving the correctness of safety protocols, rather than proving correct behavior of devices.

In this paper, we demonstrate our method by developing a three layered model of the classical block control system [10, 1, 2] and proving that it is safe. This model assumes that signals work as prescribed and the system is supported by some sort of emergency braking mechanism that ensures that a train that passes a red signal will come to a stop before it traverses another block. The three layers of the model are the following.

- A basic model defines basic concepts related to track and switching, as well as the concepts of safe state and safe operation (safe state transition) of a rail system. (Section 3.)
- An intermediate model that adds the concept of controlling the direction that trains may move on a bidirectional track sector. (There must be a switch between any two trains moving in opposite directions.) (Section 4.)
- A full-blown block control model. (Section 5.)

Each level is a state machine-based safety model, as described in Section 2. The model at each level refines its predecessor in a sense defined in Section 2. That is, each level adds detail in a way that preserves the way its predecessor operates and has a safety property that is at least as strong.

Our method can be adapted to support proof of a variety of properties implying safety as defined above. In the block system, the fundamental safety property is roughly that no two trains ever occupy the same block. This property is enforced by the fail-safe equipment that ensures that unless a train has a vacant block ahead of it, will be undergoing emergency braking and will come to a stop before entering the next block.

Emergency braking is, however, undesirable. It will occur only if a train passes a red signal. A second important safety property, therefore, is the property of normal operation: if the signals are obeyed, then emergency braking will not occur.

Most of our discussion of the block system is devoted to the fundamental safety property, but in Section 5.11 we discuss how to adapt our safety proof to show that if trains observe yellow signals by braking to a stop within the following block, then emergency braking will never occur.

The work most closely related to ours is that of Hansen [3] and of King [4]. Each builds a track model with some similarities to ours and discusses switches and signals, but is interested in simulation rather than proving theorems about safety. Our switch model is novel and is different from Hansen's.

King's paper is rather short and sketchy, and he seems to intend only a specification of signaling and not a safety proof. We are not aware of any other work that supports proofs of safety of the kind that we give. Simpson [9] presents a CSP specification of the communications necessary to support train control by wayside controllers rather than signals. If we were to refine our model further in order to define a protocol governing how sector direction is controlled, we would probably use a communication model like Simpson's.

The best-known project applying formal methods to train control, for SACEM [7, 6], appears to have concentrated on verifying the software and hardware used to implement the control system rather than on proving safety of the overall control scheme. The work in this paper is complementary. We take proper operation of devices as given and concentrate on the system aspects of safety, that is, on how the parts work together to reach the global goal of safety.

Our model, or, more properly, family of models, is based on the concept of a state machine that is commonly used in systems modeling. In the model, a rail system has at any moment a *state* consisting of (discretized) position of trains, settings of switches, aspects of signals, and possibly other things. The state of the rail system may from time to time change according to one of a set of permitted *state transitions*. Some of states are classified as *hazardous*. Those must be avoided. A way to do that is to find a set of *safe* states, disjoint from the set of hazardous states, that is invariant under state transitions (a state transition from a safe state always leads to a safe state). Then whenever the system is started in a safe state, it will always remain in a safe state.

Our basic safety concept is that in any state of the rail system, each train has an associated *safe area* such that at any time, even in an emergency situation:

- the safe area of each train contains the track occupied by that train;
- the safe area of each train never includes switches set so that the train cannot safely cross them; and
- the safe areas of distinct trains do not overlap.

Implicitly, our models all assume that each train can always brake to a stop before leaving its safe area. We do not model the fail-safe devices, trip-stops, braking profiles, etc., that may actually be used to enforce this requirement. In fact, notions of distance, time, and speed are not represented in our model, so we cannot describe how such devices would work without adding further levels that introduce these notions. Our model describes how the control scheme works, not how individual trains and switches are controlled.

At the heart of our model is a generic, abstract notion of safety that can be instantiated in order to prove safety of many specific control schemes. It is useful to separately state the top-level safety model because it identifies a set of concepts and properties that are implicitly used to ensure safety in a wide variety of control systems. Thus, when a new control system is designed or existing systems combined, the designers can check safety by showing that the essential concepts of our model can be defined in their system in such a way that the properties required by our model are satisfied. We believe that our method can also provide a basis for algorithms that will automatically check safety in specialized, well-defined contexts, such as design of signaling protocols.

The significance of our work is not so much that we can prove safety of a classical block control system; it is that we have an approach to railway safety modeling that can be used in establishing the correctness of other control methods.

We have formalized most of the concepts discussed in this paper in the specification language of the PVS theorem prover [8] and have proved numerous lemmas about those concepts, but we have not yet formalized or proved in PVS the safety theorem for block control, Theorem 5.9. Many of the proofs are not deep but require some attention to detail. Such proofs are good candidates for formal checking, but finishing the task of carrying this out will have to wait for another opportunity. Our purpose of formalizing as much as we did was to help choose the clearest of several possible formulations of some of the concepts needed to define the rules of block control, and to check those formulations by proving various lemmas about them.

2 State Machines, Safety Models, and Refinement

A *state machine* is a pair $\langle \Sigma, \vdash \rangle$ where Σ is a set, the set of *states*, and $\vdash$ is a binary relation on Σ, the *transition relation*, written infix, $S \vdash S'$. If $S \vdash S'$, we refer to (S, S') as a state transition. For all our state machines, "no change" will be a valid state transition; that is, for all $S \in \Sigma$, $S \vdash S$.

A *safety model* is a triple $\langle \Sigma, \vdash, \mathcal{S} \rangle$ where $\langle \Sigma, \vdash \rangle$ is a state machine and $\mathcal{S} \subseteq \Sigma$ is the set of *safe states*. We say that $\langle \Sigma, \vdash, \mathcal{S} \rangle$ is *safe* if $\mathcal{S}$ is invariant under state transitions: for all $S \in \mathcal{S}$ and $S' \in \Sigma$ such that $S \vdash S'$, $S' \in \mathcal{S}$.

A *refinement* of a safety model $\langle \Sigma_1, \vdash_1, \mathcal{S}_1 \rangle$ consists of another safety model $\langle \Sigma_2, \vdash_2, \mathcal{S}_2 \rangle$ together with a mapping $\rho : \Sigma_2 \rightarrow \Sigma_1$ such that

- $\rho[\mathcal{S}_2] \subseteq \mathcal{S}_1$ and
- for all transitions $S \vdash_2 S'$ with $S \in \mathcal{S}_2$, $\rho(S) \vdash_1 \rho(S')$.

The idea of a refinement is that $M_1 = \langle \Sigma_1, \vdash_1, \mathcal{S}_1 \rangle$ represents a more abstract model of a safe system, $M_2 = \langle \Sigma_2, \vdash_2, \mathcal{S}_2 \rangle$, a more detailed one. The existence of a refinement mapping ρ guarantees that the safety notion $\mathcal{S}_2$ really embodies the simpler idea $\mathcal{S}_1$, in spite of the greater detail in the model M_2. A natural way to develop a safety model is to start with a simple, very abstract model in which the notion of safety is particularly clear, then proceed through successive refinements until a sufficiently realistic one is obtained.

One other point: since unsafe states are not really of interest in proving that one model refines another, why include unsafe states in the model at all? The most important reason is to make models adaptable to different notions of safety. For example, in this paper, we use two variants of the same basic model to prove two different safety properties of the block control system.

Our definition of a safety model is not the most general one. More generally, one is given a set of hazardous states $\mathcal{H}$ (or the complement of such a set) whose complement need not be invariant. In order to run the system safely, one must find an invariant set of states $\mathcal{I}$ disjoint from $\mathcal{H}$. If one starts the system in $\mathcal{I}$, it will never enter a hazardous state. The invariant $\mathcal{I}$ corresponds to what we call the safe states of a safe model. This definition is a bit more general because one may be able to choose among several invariant sets for a given set of nonhazardous states. Furthermore, the set of hazardous states will normally be easier to define than an invariant set of safe states.

We will stick to our more specific definition of a safety model because the main focus of this paper is the refinement of safe models. It will, however, be useful to define the hazardous states in our most basic model in order to show that our notion of safety is a useful one. The definition of a hazardous state is the same in all our models.

3 The Basic Railway Safety Model

This section defines a basic mathematical model of track, trains, their states, how their states may change, and what it is for them to be in a safe state. The last three items form a safety model, which defines the basic notion of railway safety. The track model and train model together form what we call a *rail system*.

We emphasize that this is an abstract model that defines the structures essential to our style railway safety model. It can be adapted to model safety in a wide variety of railway control systems, but it does not in itself model any realistic railway control system. Rather, it is a way of stating the essential requirements that a railway control system must satisfy in order to be safe.

The essential concepts elaborated in this model are: a graph-theoretic model track, switches and their states, position of trains, the safe area of a train (where it can safely go), what safety is, and permissible transitions of train and track states.

The model is abstract because it contains few details of rail system operation beyond those necessary to mathematically represent the concepts just mentioned. *How* a railway control system maintains safety must be defined for a particular control system.

3.1 Summary of the Basic Model

The entire basic model constitutes a *rail system*. It consists of a *railnet* and a set of trains. The railnet and each train has a state, which together form the state of the rail system.

The railnet is a graph-theoretic model of the track in the rail system. It consists of a set of *track units*, complexes of pieces of track, called *edges*, that can only be traversed in certain combinations. A state of the railnet, or *netstate*, consists of a selection from each track units of a set of edges that can be safely traversed at the same time.

Trains are simply abstract objects with an associated state. A state of a train consists of its *position*, essentially the set of edges it occupies and its direction; and its *safe area*, the collection of edges that a train may at present safely occupy.

A system state is *hazardous* if two trains have overlapping positions or the position of some train is not contained in the netstate (derailment).

A system state is *safe* if the following conditions hold.

- The position of each train is contained in its safe area.
- The safe area of each train is contained in the netstate.
- Safe areas of trains are disjoint.

The safe states form an invariant set of nonhazardous states. Hazardous states will be defined in the same way in all our models, but the notion of a safe state will be progressively refined as more control structure is added to the models.

3.2 Concepts from Graph Theory

Our railway models are based on graph theory, the abstract mathematical theory of points, called *nodes* or *vertices*, and connections between them, called *edges*. The general idea is that edges represent sections of track and vertices are places where those sections of track meet. We will describe the application of this idea in more detail below. In this section, we will describe the basic graph-theoretic concepts in the abstract.

Unlike Hansen [3], we use undirected graphs, rather than directed graphs to represent track. Properly speaking, our graphs are multigraphs, because we permit more than one edge to join a given pair of vertices. We do want to specify whether the track may be traversed in only one direction or in both, but we prefer to do that with a separate function on edges giving their directionality. The reason for this preference is to stress the correspondence between edges of the graph and physical pieces of track: even though an edge joining vertices u and v may be traversed either $u \longrightarrow v$ or $v \longrightarrow u$, we want to stress that there is only one edge joining u and v. Directionality is also irrelevant to our model of interlocking switches as complexes, so it is preferable for directionality to be a separate concept rather than inherent in the notion of an edge.

For us, a graph is a quadruple (V, E, I, D). V is the set of *vertices*, E the set of *edges*. For $e \in E$, $I(e)$ is a two-element subset of V. Its elements are said to be *incident to* e, and vice versa. For $e \in E$, $D(e)$ is either $\{(u,v)\}$, $\{(v,u)\}$ or $\{(u,v),(v,u)\}$, where $I(e) = \{u,v\}$. We say that e is bidirectional if $D(e)$ has two elements. If e is unidirectional, then traffic may never traverse it except in the one permitted direction. If $(u,v) \in D(e)$ then we call u an *entry node* and v an *exit node* of e. Two edges are adjacent if they have a common incident vertex.

In our model, V represents a collection of points on a rail network. The edges represent non-overlapping sections of track connecting elements of V. If $I(e) = \{u,v\}$ then e connects u and v. The set $D(e)$ represents the directions in which e may be traversed. If $(u,v) \in D(e)$, then e may be traversed going from u to v, etc. Zero, one, or more than one edge may connect a pair of vertices.

A *directed edge* is a pair $d = (e, (u, v))$ where $(u, v) \in D(e)$. The *reversal* of a directed edge $(e, (u, v))$ is

$$reverse(e, (u, v)) = (e, (v, u)),$$

which is a directed edge if and only if e is bidirectional.

Here are the basic graph-theoretic concepts we need.

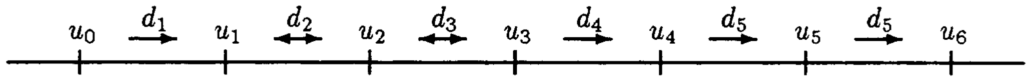
- A (directed) *path* in a graph is a sequence of directed edges

$$(e_0, (v_0, v_1)), (e_1, (v_1, v_2)), \dots, (e_n, (v_n, v_{n+1})) \tag{3.1}$$

  such that each successive edge starts from the vertex where the previous one ends.

  We will always assume that a path has at least two vertices and one edge, that is, $n \geq 0$.

  The path (3.1) is *simple* if the vertices $v_0, \dots, v_{n+1}$, and hence the edges $e_0, \dots, e_n$, are distinct. The path (3.1) is *locally simple* if for $i = 1, \dots, n-1$, $e_{i-1} \neq e_{i+1}$.



  We will also consider paths $\dots, d_{-1}, d_0, d_1, \dots$ that are potentially infinite in each direction.

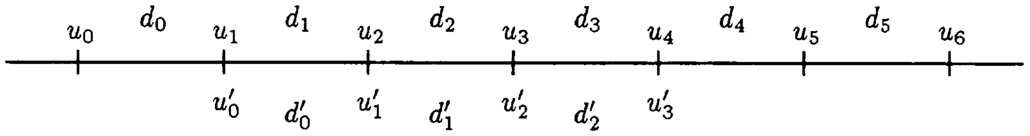
- A sequence like (3.1) is a *pseudopath* (undirected path) if for each i, e_i is an edge with distinct incident vertices v_i and v_{i+1}, but is not necessarily traversable in the direction $v_i \longrightarrow v_{i+1}$.

- We say that e is a common edge of the path in (3.1) and the path

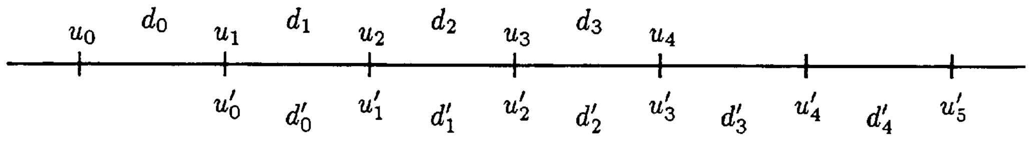
$$(e'_0, (v'_0, v'_1)), (e'_1, (v'_1, v'_2)), \dots, (e'_n, (v'_m, v'_{m+1}))$$

  if for some i and j, $0 \leq i \leq n$, $0 \leq j \leq m$, and $e_i = e'_j = e$. Similarly for pseudopaths.

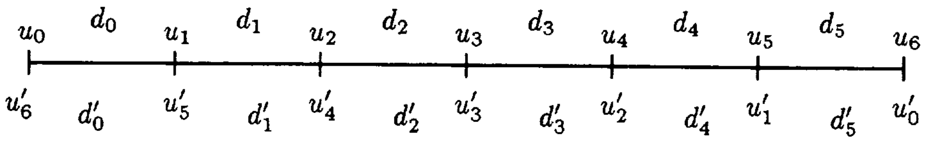
- A path $d_0, \dots, d_m$ is a subpath of a path $d'_0, \dots, d'_n$ if $d_0, \dots, d_m$ is a subsequence of $d'_0, \dots, d'_n$.



- We say that a path $d'_0, \dots, d'_n$ *advances* a path $d_0, \dots, d_m$ if there is a final subsequence of $d_0, \dots, d_m$ that is an initial subsequence of $d'_0, \dots, d'_n$



- The *reversal* of a path $d_0, \ldots, d_m$ is just the sequence *reverse*$(d_m), \ldots,$ *reverse*(d_0), which is a path if and only if each edge in the original path was bidirectional.



- "Subpath" and "advances" both include the possibility that the two paths are the same.

3.3 Track Units

The aggregate of track in the rail system is modeled as a graph. The edges are sections of track and the vertices are points dividing or terminating the edges. Just how the track is divided up into edges is rather arbitrary. We assume only that edges are short enough that the control system needs to know the position of a train only up to the edges it occupies.

Not all track edges can be safely traversed at any given time. Furthermore, only certain combinations of edges can be safely traversed at the same time. To reflect this interdependency, we group edges into *track units*. Track units are complexes of edges (and their associated vertices) whose traversability depends on each other, but not the traversability of edges of other track units. We will regard the track in a rail system not just as a collection of vertices and edges, but as a collection of track units, called the *railnet*. In our model:

- The vertices are exactly the points where *track units* (defined below) meet or end.
- The edges are simple lines of track within a track unit, going from one vertex to another without passing through any intervening vertex.

In graph-theoretic terms, a track unit is a pair $T = (N_T, \mathcal{S}_T)$ where

- N_T is a set of vertices, a subset of the set of all vertices.
- $\mathcal{S}_T$ is a set of sets of edges connecting the vertices (set of subsets of E all of whose members connect pairs of members of N_T).
- Each set of edges $S \in \mathcal{S}_T$ is called a *state* of the track unit.
- For each state $S \in \mathcal{S}_T$, each vertex $p \in N$ is incident to at most one edge $e \in S$.
- If $\mathcal{S}_T$ contains more than one state, then $\emptyset$ (the empty set of edges) is also a state in $\mathcal{S}$.
- Two states S_1, S_2 of a track unit are said to be *mutually accessible* if either $S_1 = S_2$ or one of S_1 or S_2 is the empty set.

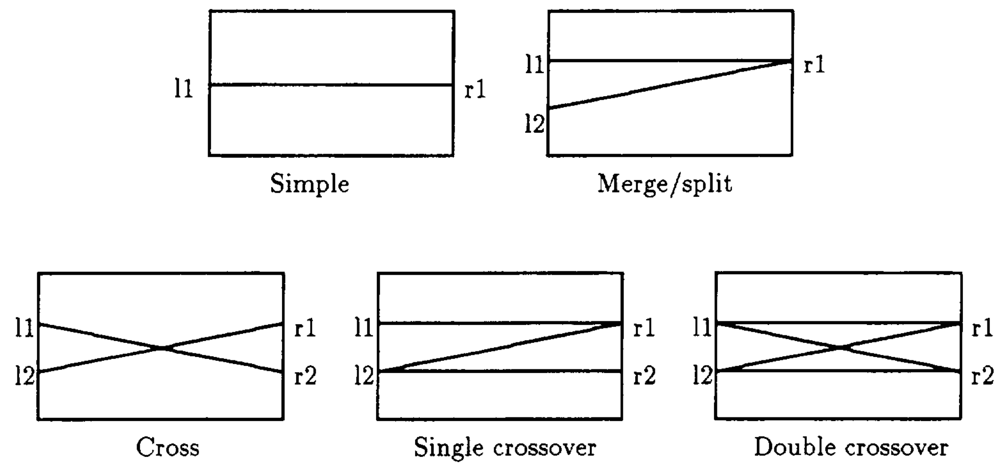


Figure 1: Five kinds of track units.

- We say that an edge in one of the states of a track unit *occurs* in the track unit.

In a track unit $(\mathcal{N}_T, \mathcal{S}_T)$, the members of $\mathcal{S}_T$ are sets of edges that can be safely, simultaneously traversed.

The most common kinds, perhaps the only kinds, of track units that occur are those in Figure 1. A track unit isomorphic to the one labeled *simple* is called a *simple* track unit. All others are called *complex* track units or *switches*.

A complex track unit reflects the idea of an interlocking, that is, a complex of short pieces of track and related switches that are opened and closed in a coordinated manner. Although the term switch applies more properly to a node at which more than two edges meet, we will follow colloquial practise and apply the term to a complex track unit.

The possible states of a switch reflect the combinations in which the interlocking logic permits its edges to be traversed. The idea of the accessibility relation is that to change switch settings, it is necessary first to open the switch, that is, change the state of the track unit to the empty set (no safe traversals possible).

Note that our model of track units and their states relates only to the underlying undirected graph, not to the directionality. An edge is either in or out of a state. It cannot be in with one direction and out with the other.

To understand the track unit model, let us see how to use it to represent the track units in Figure 1. Specifically, let us consider the most complex track unit in Figure 1, the double crossover switch. We denote, say, the edge joining `l1` and `r1` by `l1-r1`. In the double crossover switch,

$$\mathcal{N} = \{\texttt{l1, l2, r1, r2}\}$$
$$\mathcal{S} = \{\emptyset, \{\texttt{l1-r2}\}, \{\texttt{l2-r1}\}, \{\texttt{l1-r1}, \texttt{l2-r2}\}\}$$

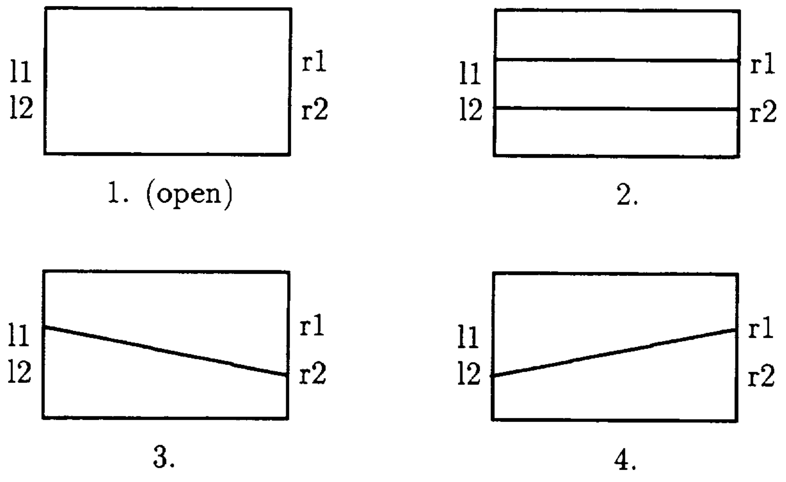


Figure 2: Possible settings of a double-crossover switch.

The members of N are the vertices where the switch joins other track units. The states of the switch correspond respectively to the four possible settings of the switch, shown in Figure 2:

1. open (cannot be safely traversed);
2. can be safely (simultaneously) traversed along the lines `l1-r1` and `l2-r2`.
3. can be safely traversed only along the line `l1-r2`;
4. can be safely traversed only along the line `l2-r1`;

Two states are mutually accessible if the corresponding switch setting can be changed directly from one to the other. Thus, a non-open setting can be changed to another non-open setting only by first opening the switch. An open switch can be changed to any non-open setting. In state machine terms, each track unit $T = (N_T, \mathcal{S}_T)$ induces a state machine

$$\langle \mathcal{S}_T, \vdash_T \rangle$$

where for $E, E' \in \mathcal{S}_T$,

$$E \vdash E' \iff (E = E') \vee (E = \emptyset) \vee (E' = \emptyset).$$

We conclude by giving the models for each of the five kinds of track unit given in Figure 1.

- Simple. $N = \{$`l1,r1`$\}$, $\mathcal{S} = \{\{$`l1-r1`$\}\}$.
- Merge/split. $N = \{$`l1,l2,r1`$\}$, $\mathcal{S} = \{\emptyset, \{$`l1-r1`$\}, \{$`l2-r1`$\}\}$.

- Cross. $N = \{\texttt{l1}, \texttt{l2}, \texttt{r1}, \texttt{r2}\}$, $\mathcal{S} = \{\emptyset, \{\texttt{l1-r2}\}, \{\texttt{l2-r1}\}\}$.
- Single crossover switch. $N = \{\texttt{l1}, \texttt{l2}, \texttt{r1}, \texttt{r2}\}$, $\mathcal{S} = \{\emptyset, \{\texttt{l1-r1}, \texttt{l2-r2}\}, \{\texttt{l2-r1}\}\}$.
- Double crossover switch.

$$N = \{\texttt{l1}, \texttt{l2}, \texttt{r1}, \texttt{r2}\},$$
$$\mathcal{S} = \{\emptyset, \{\texttt{l1-r1}, \texttt{l2-r2}\}, \{\texttt{l2-r1}\}, \{\texttt{l1-r2}\}\}.$$

We remark that the states of a switch reflect its logical behavior, not its physical behavior. For example, a cross typically does not contain any switching mechanism. Rather, the control system must be such that the two edges are never traversed at the same time. In the classical block control system with signals, signals must forbid entry to at least one edge of the cross at any time.

Similarly, the author has been assured that it is physically possible to set a double crossover switch so that it functions as a cross. Either the switch setting mechanism must never actually set it that way, or else the control system must, as for the cross, permit trains to cross only one diagonal at a time.

We could make a slightly fancier model of a simple track unit by letting it have two states: one consisting of its single edge, and the other the empty state. The empty state would represent a broken rail. No changes to the basic model would be required by this to support this more general model of simple track, and only minor changes to the models we discuss later.

3.4 Model of a Railnet

The *railnet* is a collection $\mathcal{T}$ of track units. We will write $T = (N_T, \mathcal{S}_T)$ for $T \in \mathcal{T}$ a track unit, where N_T is the set of vertices of T and $\mathcal{S}_T$ is the set of states of T. We also write E_T for the set of edges occurring in T, $E_T = \bigcup_{S \in \mathcal{S}_T} S$. Assigning a state (switch setting) to each track unit in the railnet will produce a global *state* of the railnet, the collection of all edges that can be safely traversed with those switch settings.

Different track units may share vertices. In fact, we assume that the vertices of a railnet are partitioned into two kinds, terminal and nonterminal vertices.

- Terminal vertices belong to exactly one track unit.
- Nonterminal vertices belong to exactly two track units.

We further assume that:

- no two track units share an edge;
- each vertex of a switch is nonterminal; and
- no two switches share a vertex.

The first requirement reflects an essential property of a safe rail system. The last two requirements are not essential, but any rail system can be modeled so that they are true, and they make things simpler. The significance of the second requirement will become clear later in Section 5.

With the railnet $\mathcal{T}$ is associated a graph called the *supernet*, (N, E), given by

$$N = \bigcup_{T \in \mathcal{T}} N_T, \; E = \bigcup_{T \in \mathcal{T}} E_T.$$

That is, the set of vertices of the supernet consist of all vertices of any track unit in the railnet, and the set of edges consists of all edges occurring in any track unit in the railnet.

A *state* of the railnet, or *netstate*, is a set of edges $S_{\text{net}} \subseteq E$ such that for each track unit T,

$$S_{\text{net}} \cap E_T \in \mathcal{S}_T. \tag{3.2}$$

The netstate represents the ensemble of all track edges that can be safely traversed a given time.

Netstates correspond to another natural notion of state of the railnet, namely an assignment of a state $S_T \in \mathcal{S}_T$ to each track unit T. We call such an assignment a *track state assignment*. Physically, it prescribes a particular setting for each switch in the rail net.

Each netstate S_{net} induces a track state assignment given by

$$S_T = S_{\text{net}} \cap E_T, \; T \in \mathcal{T}. \tag{3.3}$$

Conversely, each track state assignment S_T, $T \in \mathcal{T}$, induces a netstate S_{net} by

$$S_{\text{net}} = \bigcup_{T \in \mathcal{T}} S_T. \tag{3.4}$$

In each case, S_{net} is the set of track edges that can be safely traversed with the switch settings indicated by the track assignment S_T, $T \in \mathcal{T}$.

Because track units do not share edges, S_{net} as defined in eq. (3.4) satisfies the condition in eq. (3.2) and is therefore a netstate. Furthermore, eqs. (3.3) and (3.4) define inverse operations. Thus, netstates and track state assignments stand in one-to-one correspondence. They represent two equivalent definitions of railnet state, by traversable edges and by switch settings.

The proper notion of transition for railnets is indicated by the correspondence with track assignments. That is, a netstate transition from S_{net} to S'_{net} is possible if for each track unit $T \in \mathcal{T}$, the state S_T of T induced by S_{net} and the state S'_T of T induced by S'_{net} are mutually accessible.

Formally, the netstates and their transitions form a state machine

$$\langle \Sigma_{\text{net}}, \vdash_{\text{net}} \rangle.$$

Σ_{net} is the set of all netstates S_{net} and

$$S_{\text{net}} \vdash_{\text{net}} S'_{\text{net}} \iff \forall T \in \mathcal{T}, \; S_T \vdash_T S'_T,$$

where S_T, $T \in \mathcal{T}$, and S'_T, $T \in \mathcal{T}$, are respectively the track assignments induced from S_{net} and S'_{net} via eq. (3.3).

Each netstate S_{net} induces a subgraph (N, S_{net}) of (N, E). We will often speak as if S_{net} were this subgraph. This subgraph is of degree two; therefore each of its components must be either a path or a cycle.

Theorem 3.1 *For any netstate S_{net}, any vertex is incident to at most two edges of S_{net}.*

Proof. Each vertex is a node of at most two track units and any netstate contains at most one edge incident to each node of each track unit. □

3.5 Trains, Positions, and Safe Areas

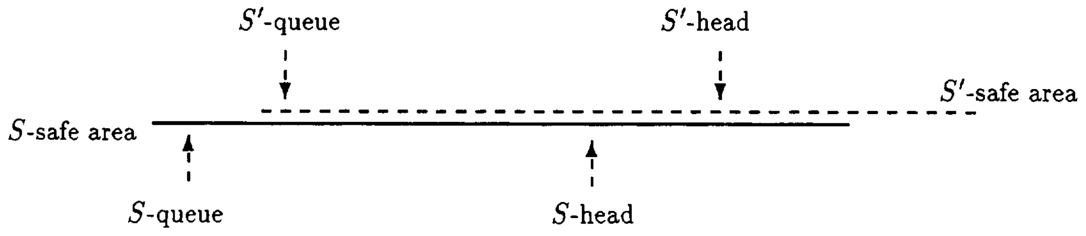
From the point of view of our model, a train is simply an abstract entity known by its state, which at this level consists of its *position* and its *safe area*.

- the position of a train is a simple path in the supernet, representing the set of edges that the physical position of the train overlaps. If this path is $(e_0, (v_0, v_1)), \ldots, (e_n, (v_n, v_{n+1}))$, then the train is considered to be moving in the direction v_0 to v_n (even if the physical train would be standing still). The vertex v_{n+1}, is called the *head* or *head vertex* of the train; e_n is called the *head edge*; v_0 is called the *queue*.
- the safe area r of a train τ is a simple path in the supernet, representing an area of track that τ currently has permission to occupy.
- the position must be a subpath of the safe area.

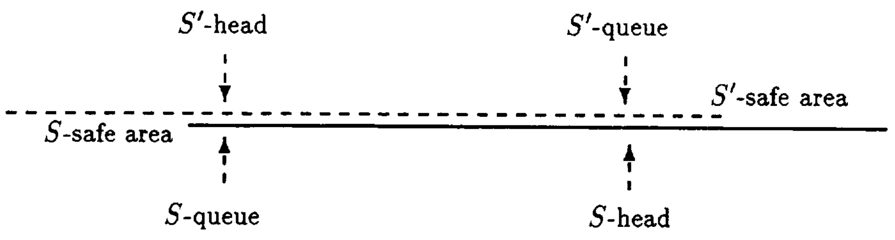
In this model, head and queue need not pinpoint the actual head and queue position of the physical train. Rather, they are the vertices in front of the actual head and behind the actual queue position of the train.

A transition from a train state s to a train state s' is permitted in the following two cases.

- Normal progress. the s'-position advances the s-position, the s'-safe area advances the s-safe area, and the s-position is a subpath of the s'-safe area.



- Reversal. The S'-position reverses the S-path. The S'-safe area can be any path that has the S'-position as a subpath.



The idea of reversal is that a train comes to a stop, reverses its intended direction of travel, has a new safe area designated, then proceeds in the new direction.

Putting all this together, train states and their transitions define a state machine

$$\langle \Sigma_{\textbf{train}}, \vdash_{\textbf{train}} \rangle$$

where

- $\Sigma_{\textbf{train}}$ is the set of pairs $(\textit{position}, \textit{safe-area})$ where *safe-area* is a path in the supernet and *position* is a subpath of *safe-area*, and
- $(\textit{position}, \textit{safe-area}) \vdash_{\textbf{train}} (\textit{position}', \textit{safe-area}')$ if either
  - *position*$'$ advances *position*, *safe-area*$'$ advances *safe-area*, and *position*$'$ is a subpath of *safe-area*, or
  - *position*$'$ is the reversal of *position*.

3.6 The Rail System

A *rail system* consists of a railnet $\mathcal{T}$ and a set of trains *Train*. A *rail system state* is a pair $(S_{\text{net}}, S_{\text{train}})$, where:

- $S_{\text{net}} \in \Sigma_{\text{net}}$ is a netstate (state of the railnet);
- S_{train} is a function $\textit{Train} \to \Sigma_{\text{train}}$ assigning a state to each train.

A state $(S_{\text{net}}, S_{\text{train}})$ is *safe* if

- for each train $\tau \in \textit{Train}$, the safe area of $S_{\text{train}}(\tau)$ is a subpath of S_{net}; and
- for any two trains $\tau, \tau' \in \textit{Train}$, the safe areas in $S_{\text{train}}(\tau)$ and $S_{\text{train}}(\tau')$ have no common edges.

Let Σ_{basic} be the set of rail system states and let $\mathcal{S}_{\text{basic}}$ be the set of safe states.

A system state is *safe* if the safe areas of trains are disjoint and subpaths of the netstate.

Because each train is in its safe area, the safe areas pass only through safe switches, and safe areas are simple paths and do not intersect other safe areas, it follows that the trains cannot collide in a safe state.

3.7 State Transitions

Our intention is that the state transitions in our model should be like the state transitions that would occur in a physical rail system. That is, the state changes whenever the safe area of a train changes, whenever the head or queue of a train crosses an edge boundary, whenever a train changes direction, and whenever the state of a switch changes.

Let $S = (S_{\text{net}}, S_{\text{train}})$ and $S' = (S_{\text{net}}, S_{\text{train}})$ be states of the rail system. A state transition $S \vdash_{\text{basic}} S'$ is permitted if the following conditions hold.

- $S_{\text{net}} \vdash_{\text{net}} S'_{\text{net}}$.
- For each train τ, the following conditions hold.
  - $S_{\text{train}}(\tau) \vdash_{\text{train}} S'_{\text{train}}(\tau)$.
  - Either *safe_area*$'(\tau)$ is a path in S'_{net} and shares no edges with safe areas of other trains, or else *safe_area*$'(\tau) =$ *safe_area*(τ).

Now $\langle \Sigma_{\text{basic}}, \vdash_{\text{basic}}, \mathcal{S}_{\text{basic}} \rangle$ defines a state machine induced by the rail system. Our definitions make the following Theorem trivial.

Theorem 3.2 $\langle \Sigma_{\text{basic}}, \vdash_{\text{basic}}, \mathcal{S}_{\text{basic}} \rangle$ *is safe.*

It is possible to drop the restriction on train state transitions that the path can advance only one edge. We impose it only in order to conform to the idea that a state change should occur whenever the model's representation of the state of the physical system would change, e.g., when ever the physical head or queue of a train crosses a vertex.

4 Sectors and Direction Control

The basic model states that the safe areas of distinct trains may not overlap. But how is this requirement to be enforced? One way is to separate it into two parts: trains moving in opposite directions and trains moving in the same direction. In this section, we will address the first part by dividing the netstate into sectors each of which may intersect only the safe areas trains traveling in a particular direction.

In the next section, we will present the signaling mechanisms that enforce this direction control as well as the condition that safe areas of trains moving in the same direction do not intersect.

The key notion of this section is that of a *sector*. A sector is a stretch of track between switches. The state of a sector is the direction in which it may currently be traversed. The safe area of a train may share nodes only with sectors whose direction is compatible with the safe area. This condition implies that safe areas of trains moving in opposite directions cannot overlap.

4.1 No-Cycle Condition and Flow Condition

We will assume that our track satisfies two natural conditions.

> **No-cycle condition.** The supernet contains no simple cycles of edges of simple track units.

From this condition and the finiteness of the set of edges, it follows that any pseudopath of simple edges of simple track units that is locally simple (does not double back on itself) is finite.

> **Flow Condition at a Vertex.** Any vertex v must be both an entry node of some edge e_1 and an exit node of some edge e_2. If v belongs to more than one track unit, then it must be possible to choose such e_1 and e_2 belonging to different track units.

Figure 3 shows a number of examples in which the flow condition does or does not hold. The flow condition permits track to cease being bidirectional only by splitting directions at a switch or by ending altogether.

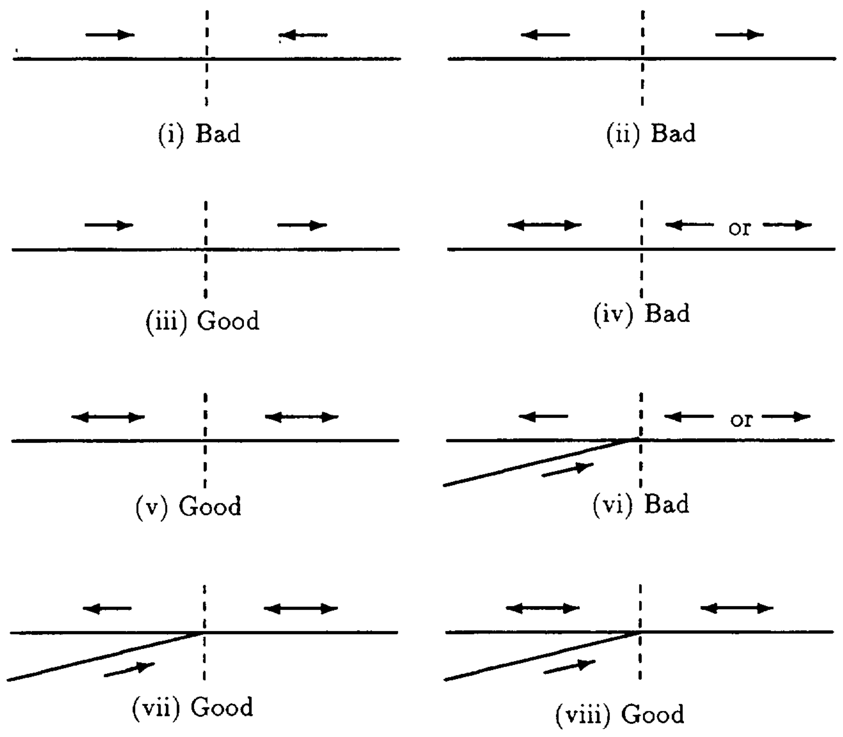
The flow condition implies that every locally simple pseudopath of edges of simple track units is a path or the reversal of a path.

4.2 Sectors

A *sector* C is a maximal set of contiguous edges of simple track units. That is:

- C is nonempty;
- all members of C are edges of simple track units;
- any edge of a simple track that is adjacent to an edge of C belongs to C;
- any two edges in C are joined by a pseudopath of edges in C.

In other words C is an edge-component of the subgraph of the supernet whose edges are the edges of simple track units.



Dashed line indicates track unit boundary.

Figure 3: Flow through a vertex: permissible and forbidden configurations.

Because any vertex is incident to at most two edges of simple track units, each sector must consist of the edges of a simple pseudopath

$$(e_0,(v_0,v_1)),(e_1,(v_1,v_2)),\ldots,(e_n,(v_n,v_{n+1})) \tag{4.5}$$

By the no-cycle condition, v_0 and v_{n+1} are distinct and not incident to any other edge of a simple track unit. By the flow condition, the pseudopath (4.5) is a path or the reversal of a path (both, if any of its edges is bidirectional). The nodes $v_0,\ldots,v_{n+1}$ are the nodes of the sector C. Because sectors are closed under adjacency, distinct sectors have disjoint node sets. Because terminal nodes belong only to simple track units and switches are adjacent only to simple track units, every node belongs to some sector, called the sector of that node. Likewise every edge of a simple track unit is the edge of a unique sector.

We can regard a sector like sector C in (4.5) as a kind of big edge and apply the terminology of edges to sectors. The incident nodes of C are v_0 and v_{n+1}. If the pseudopath in (4.5) is actually a path then v_0 is an *entry node* of C and v_{n+1} is an *exit node* of C; if its reversal is a path, then v_{n+1} is an entry and v_0 is an exit.

A *direction* of a sector is a pair of vertices (u, v) where u is an entry node and v is an exit node of the sector. $D(C)$ is the set of directions of a sector C.

4.3 The Basic Model with Direction Control—Objects and States

We add sectors to the basic model.

A state of a sector C is a simple path whose set of edges is C. The states of a sector correspond to its directions, but a number of definitions are simpler if we let the state of a sector be a path instead of a pair of endpoints of the sector.

Let *Sector* be the set of sectors.

A state of the sector model is a triple

$$(S_{\text{net}}, S_{\text{train}}, S_{\text{sector}})$$

such that $(S_{\text{net}}, S_{\text{train}})$ is state of the basic model and S_{sector} is a function on sectors such that for every sector C, $S_{\text{sector}}(C)$ is a state of C.

4.4 Safe States with Sectors

Definition 4.1 *Two locally simple paths P and Q are* compatible *if either*

4.1.1 they share no vertices or

4.1.2 P and Q are both (directed) subpaths of some locally simple (directed) path R.

See Figure 4.

Definition 4.2 *The set S_{sector} of safe states of the sector model consists of all states satisfying the following conditions.*

4.2.1 For each train τ, safe_area(τ) *is a path in S_{net}.*

4.2.2 For each train τ and sector C, safe_area(τ) *is compatible with $S_{\text{sector}}(C)$.*

4.2.3 Whenever τ and τ' are distinct trains such that safe_area(τ) *and* safe_area(τ') *have a common edge e, the two safe areas orient e differently.*

Condition 4.2.2 will guarantee that trains traveling in opposite directions will not have overlapping safe area. The sector model does not, however, contain any mechanism forcing safe area of trains traveling in the same direction to be disjoint, so we just have to make it part of the definition of a safe state (condition 4.2.3).

The sector model will refine the basic model via the mapping

$$\rho : (S_{\text{net}}, S_{\text{train}}, S_{\text{sector}}) \mapsto (S_{\text{net}}, S_{\text{train}}).$$

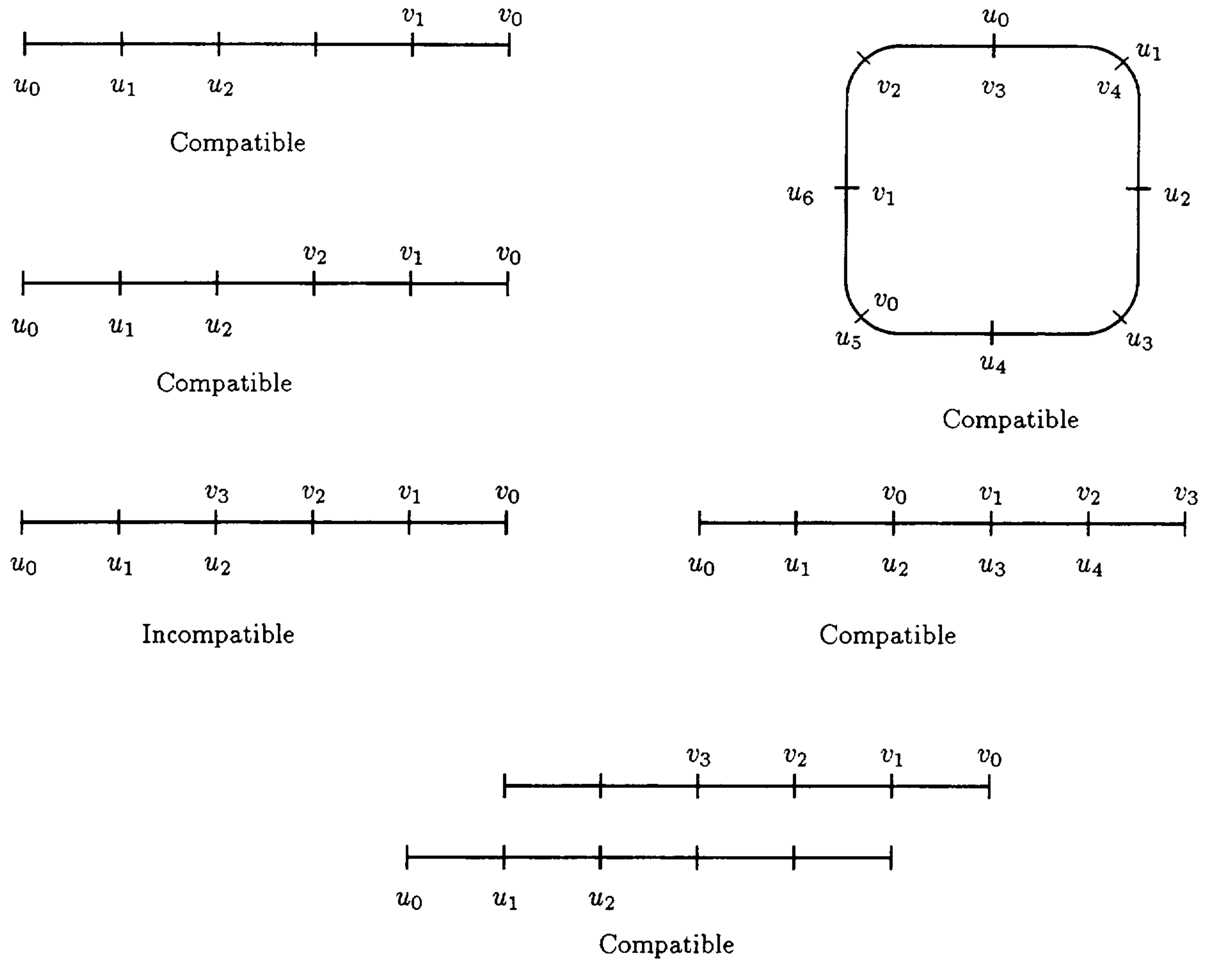


Figure 4: Compatible and incompatible paths.

Lemma 4.3 *The function ρ maps $\mathcal{S}_{\text{sector}}$ into $\mathcal{S}_{\text{basic}}$.*

Proof. We need only show that safe areas of distinct trains do not share any edges. Suppose the contrary, that τ and τ' are distinct trains such that e is an edge of both *safe_area*(τ) and *safe_area*(τ'). By 4.2.3, the safe areas of τ and τ' give e different orientations. If v is a vertex incident to e, then either *safe_area*(τ) or *safe_area*(τ') must be incompatible with the state of the sector of v.

4.5 State Transitions

For $S, S' \in S_{\text{sector}}$, $S \vdash_{\text{sector}} S'$ if the following conditions hold.

- $S_{\text{net}} \vdash_{\text{net}} S'_{\text{net}}$.
- For each train τ, the following conditions hold.
  - $S_{\text{train}}(\tau) \vdash_{\text{train}} S'_{\text{train}}(\tau)$, and
  - either *safe_area*$'(\tau)$ is a path in S'_{net} and shares no edges with safe areas of other trains, or else *safe_area*$'(\tau) =$ *safe_area*(τ).
- For any sector C, $S_{\text{sector}}(C) = S'_{\text{sector}}(C)$ unless for each train τ, if *safe_area*(τ) or *safe_area*$'(\tau)$ shares a vertex with C then *position*$'(\tau)$ is the reversal of *position*(τ).

This definition makes the mapping ρ a refinement.

5 Classical Block Control

In this section we model the classical block control system with its associated signals. Our main references for the block system are [10, 1, 2].

In a block control system, the railnet is divided into pieces called blocks. Two trains are never permitted to occupy the same block, and the speed and weight of trains are restricted so that any train can stop in the length of a block. A system of signals warns a train when it is approaching another, early enough that the train has a block in which to stop. In this section we use the term *occupy* in a technical sense defined in Subsection 5.5.

As we mentioned in the Introduction, we are primarily concerned with the fundamental safety property that trains do not occupy the same block, hence do not collide. Hence in our main discussion we do consider the possibility that a train may pass a red signal and be brought to a stop by having its emergency brakes tripped. In Section 5.11, we show how our proof can be modified to show that under normal operation, i.e., obeying yellow signals, emergency braking will never be triggered.

Many variations of block control exist. The block control system presented here has simple signals with three aspects (green, yellow, red) and controls trains so that no two trains ever occupy the same block and, except in emergency situations, trains are always separated by an empty block.

5.1 Blocks

In our model, a block is a section of track, not containing any switches, exit from which is controlled by signal, or, if the block is bidirectional, signals. Movement through switches is controlled by signals associated with adjacent blocks. If one wants to consider adjoining switches with movement from one to another controlled by a signal, then imagine a block of length zero inserted between them. (In such a case, a train entering a switch adjacent to the zero-length block must be moving slowly enough that it stop in the length of the switch edge.)

Since we are considering only pure block control, we will simply let blocks be edges of simple track units. Other kinds of control, the simplest being block control with some permissive signals (through which one may proceed slowly enough to stop on sight) can be modeled by letting blocks be collections of smaller edges, as we have modeled sectors. In our model, a red signal always indicates absolute stop, never permissive stop (proceed at reduced speed, prepared to stop on sighting a train ahead), because we do not model the possibility of more than one train occupying a block.

Since blocks are edges we apply to them the usual terminology about entry and exit nodes.

5.2 Signals

At each exit node b of a block B there is a signal $s_{B,b}$. The signal $s_{B,b}$ is considered to be physically located at b, facing toward traffic in B moving in the direction from $a \longrightarrow b$, where $I(B) = \{a, b\}$, governing when traffic is permitted to exit through node b. Figure 5 shows where signals must be placed in a variety of circumstances.

According to this rule governing signal placement, there must be a signal $s_{B,b}$ located at b even if b is a terminal node. In practise, there would probably not be a signal there, but the control system must still behave as if there were. This imaginary signal would always be red and closed according to the rules we give below.

The state of a signal is a pair consisting of an *aspect*, the color of light it is displaying.

$$\begin{aligned} \mathit{Aspect} &= \{\mathit{green}, \mathit{yellow}, \mathit{red}\} \\ \Sigma_{\mathbf{signal}} &= \mathit{Aspect} \times \{\mathit{open}, \mathit{closed}\} \end{aligned}$$

We will not explicitly consider signals that are dark (not functioning), since dark signals are treated as red. Our model assumes that standard signal fail-safe mechanisms, as described in [10], are used so that a malfunctioning signal can only be dark and can never display a false aspect.

The *indication* of a signal aspect is its meaning to a train that sees it. Indications correspond to aspects as follows:

- green—proceed (run in safety);
- yellow—approach, prepared to stop at the next signal;
- red—stop before passing the signal.

Indications define the intended normal behavior of the system, however, but such behavior cannot always be guaranteed. For example, one cannot guarantee that a train will stop before a red signal

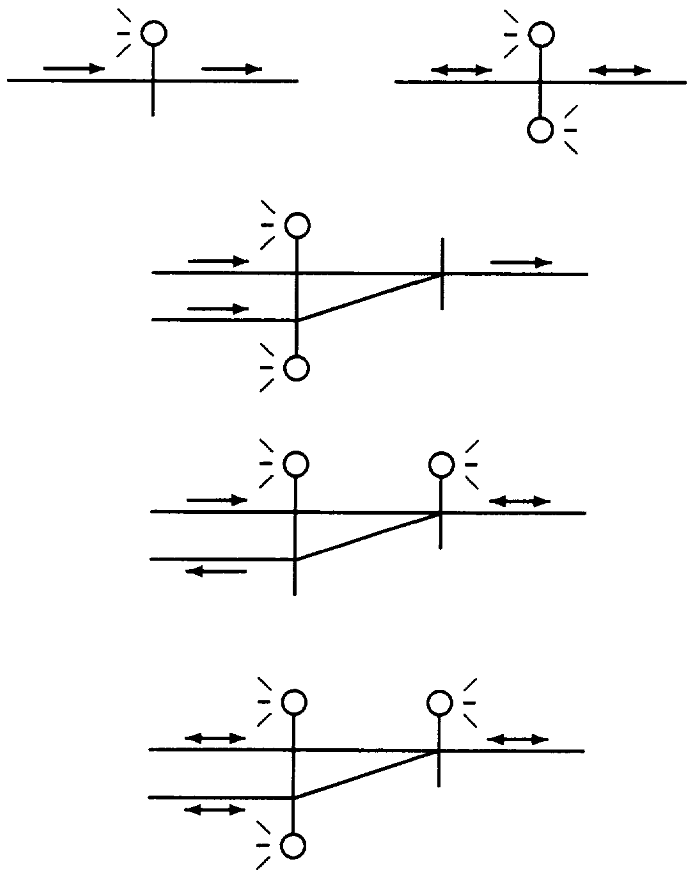

Signals "shine" toward the blocks they govern.
Vertical lines indicate block boundaries.

Figure 5: Signal placement.

unless it first passes a yellow signal warning its engineer of the red signal ahead. Now, the only way a train can approach a red signal without first passing a yellow signal is if it passes a signal just as it would have turned yellow. Certainly such coincidences will be rare, but when they occur, the train's engineer will not know to stop until the train approaches the red signal, by which time it may not be able to stop. Even if the yellow signal is seen, there is nothing actually forcing the train to stop. If a train does pass a red signal, however, trip stops or some other fail-safe mechanism will engage the train's emergency brakes, making sure that it will stop before passing the next red signal. This is why the signaling protocol we present normally keeps a vacant block between trains.

Being closed is not an observable property of a signal, but rather a concept describing a signal that a train will not pass, even in an emergency situation. A closed signal is always red. If a signal is closed, no train can approach it without first passing another red signal and being halted by trip stops triggering its emergency brakes.

In this model, yellow and green signals mean the same from the point of view of safety in the strict sense and are treated the same. The difference between them is, however, important for efficient railway operation, since it is inefficient for trains to have to emergency brake frequently. In section 5.11, we discuss ways to modify the block model so that it models normal operation in non-emergency situations, rather than safe operation even in emergencies.

5.3 State of a Train

As in the basic model and the sector model, the state of a train consists of its position and safe area.

5.4 The Block Preceding or Following a Signal or Another Block

The protocol that determines signal aspects in the block control system is defined in terms of how many blocks past the signal are accessible and unoccupied. In this section we present the graph theoretic definitions required to define which blocks are relevant to a given signal.

Assume that a netstate S_{net} is given.

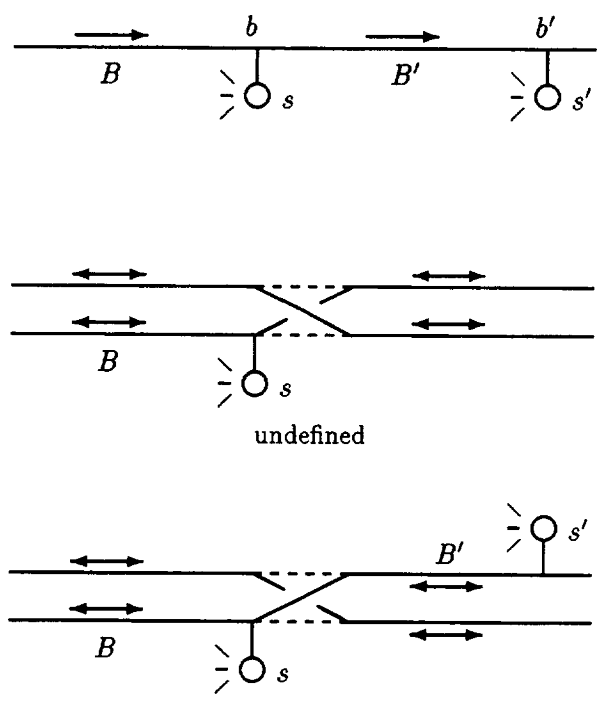
Consider an edge e of the supernet with incident nodes a and b. In S_{net}, the *directed component* of e, in direction $a \longrightarrow b$, is a maximal path $\ldots, d_{-1}, d_0, d_1, \ldots$ such that the following conditions hold.

- $d_0 = (e, (a, b))$.
- For each $n \geq 0$, if $d_n = (e_n, (v_n, v_{n+1}))$, and there exists $e_{n+1} \in S_{\text{net}} \setminus \{e_n\}$ for which v_{n+1} is an entry node, then $d_{n+1} = (e_{n+1}, (v_{n+1}, v_{n+2}))$, where v_{n+2} is the other vertex incident to e_{n+1}. If d_n does not exist or there is no such e_{n+1}, then d_{n+1} does not exist.
- For $n \leq 0$, define d_{n-1} from d_n in dual fashion.

If it exists, e_{n+1} is unambiguously defined, because in any netstate there are at most two edges to which any vertex is incident.

The edges of the component of an edge e in the direction $a \longrightarrow b$ are the edges of the directed component (in the usual sense) of $(e, (a, b))$ in the directed graph induced by (N, S_{net}) and the direction function D. Because each vertex of (N, S_{net}) has degree at most two, the component always forms a path. We simply number the edges of this path so that the zero-th edge is $(e, (a, b))$.

The nth *block following* the directed block $d_0 = (e, (a, b))$ is the nth edge e_m, $m > 0$, searching from e_1 up, that is a block. If there are not n such blocks, then the nth block following $(e, (a, b))$ is said not to exist. We define the nth block preceding $(e, (a, b))$ dually. The nth signal following



Dashed lines indicate edges not in the current track state.

Figure 6: The block and signal following a given block or signal.

(preceding) $(e, (a, b))$ is $s_{e_m, v_{m+1}}$ where e_m is the nth block following (preceding) $(e, (a, b))$. The nth block (signal) following (preceding) a signal $s_{B,b}$ is the nth block (signal) following (preceding) $(B, (a, b))$, where a is the other vertex incident to B. We call $\ldots, d_{-1}, d_0 = (B, (a, b)), d_1, \ldots$ the *component of* $s_{B,b}$.

The set of edges *governed by* $s_{B,b}$ in a netstate S_{net} consists of the edges $e_{m+1}, e_{m+2}, \ldots, e_0 = B$ where e_m is the first block preceding B in S_{net}, or, if there is no block preceding B, m is the first negative index for which d_m does not exist. In other words, $s_{B,b}$ governs B and any switch edges intervening between B and the block preceding B. In the block model, since non-switch edges are all blocks and switches are only one edge wide and do not adjoin, m is always either -1 or -2.

The various possibilities for block following are shown in Figure 6.

5.5 States of the Block Control Model

A state in the block control model consists of components giving the netstate and states of signals, trains and sectors. The safety mechanisms of the block control system impose some physical restrictions on how the components of a state must be related. We describe those restrictions in this section.

An edge e is *associated with* a block B if $e = B$ or e is an edge of a switch and is adjacent to B.

A train τ *occupies* an edges e (in a given state) in either of the following circumstances:

- e belongs to $position(\tau)$; or
- e is a block and an edge associated with e belongs to $position(\tau)$.

Definition 5.1 *The set Σ_{block} of all states of the block model consists of all quadruples*

$$S = (S_{\text{net}}, S_{\text{signal}}, S_{\text{train}}, S_{\text{sector}}) \in \Sigma_{\text{net}} \times \Sigma_{\text{signal}} \times \Sigma_{\text{train}} \times \Sigma_{\text{sector}}$$

such that for each signal s, the following conditions hold.

Here, we write

$$S_{\text{train}}(\tau) = (position(\tau), \text{safe\_area}(\tau)).$$

Similarly, for a signals s we write

$$S_{\text{signal}}(s) = (\text{aspect}(s), b),$$

where $b \in \{\text{open}, \text{closed}\}$.

5.1.1 If the second signal following s in S_{net} does not exist or the first or second block following s is occupied by a train, then aspect(s) = red.

5.1.2 If the first signal s' following s in S_{net} exists and aspect(s') = red, then aspect(s) $\in$ {yellow, red}.

5.1.3 If a signal s is not closed, then the the block B following s exists, is not occupied, and the state of the component of B is compatible with the component of s.

5.1.4 For each train τ, safe_area(τ) consists of one of the following.

(i) All edges in position(τ), plus any additional blocks occupied by τ.

(ii) The foregoing, plus all edges as far as the signal following the signal governing τ (i.e. one additional block plus any intervening edges) if both signals exist.

The safe area of τ is oriented compatibly with position(τ).

5.1.5 If s is closed, then its aspect is red and so is the aspect of the signal preceding s, if it exists.

We allow signals that are not obliged by either of these rules to be red anyway in order to accommodate malfunctioning (dark) signals, treated as red, and to permit signals to be set to red by other traffic control mechanisms.

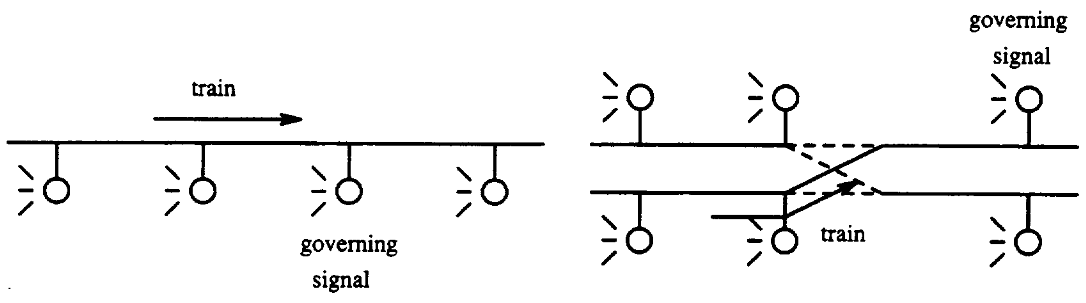


Figure 7: The signal governing a train.

5.6 Safe States for Block Control

In this section, we will define what it is for a block control state to be safe.

As mentioned in Section 2, deciding which properties to put into the definition of a state, which to put into the definition of a safe state, and how to define state transitions is somewhat arbitrary. The rule we have followed is that if our model does not provide a mechanism showing how a property is maintained, then that property should be included in the definition of a state. For example, the component of an open signal must be compatible with the state (direction) of the sector of the block following it. In practise, this property would be maintained by a protocol for setting sector directions and for giving signals permission to open. Such a protocol is not represented in our model, so we simply assume that all states satisfy this compatibility property.

Definition 5.2 *The set $\mathcal{S}_{\text{block}}$ of safe states of the control system consists of all states*

$$S = (S_{\text{net}}, S_{\text{signal}}, S_{\text{train}}, S_{\text{sector}}) \in \Sigma_{\text{block}}$$

such that the following conditions hold.

5.2.1 For each train τ, position(τ) is a subpath of the netstate S_{net} and is compatible with the state of each sector C.

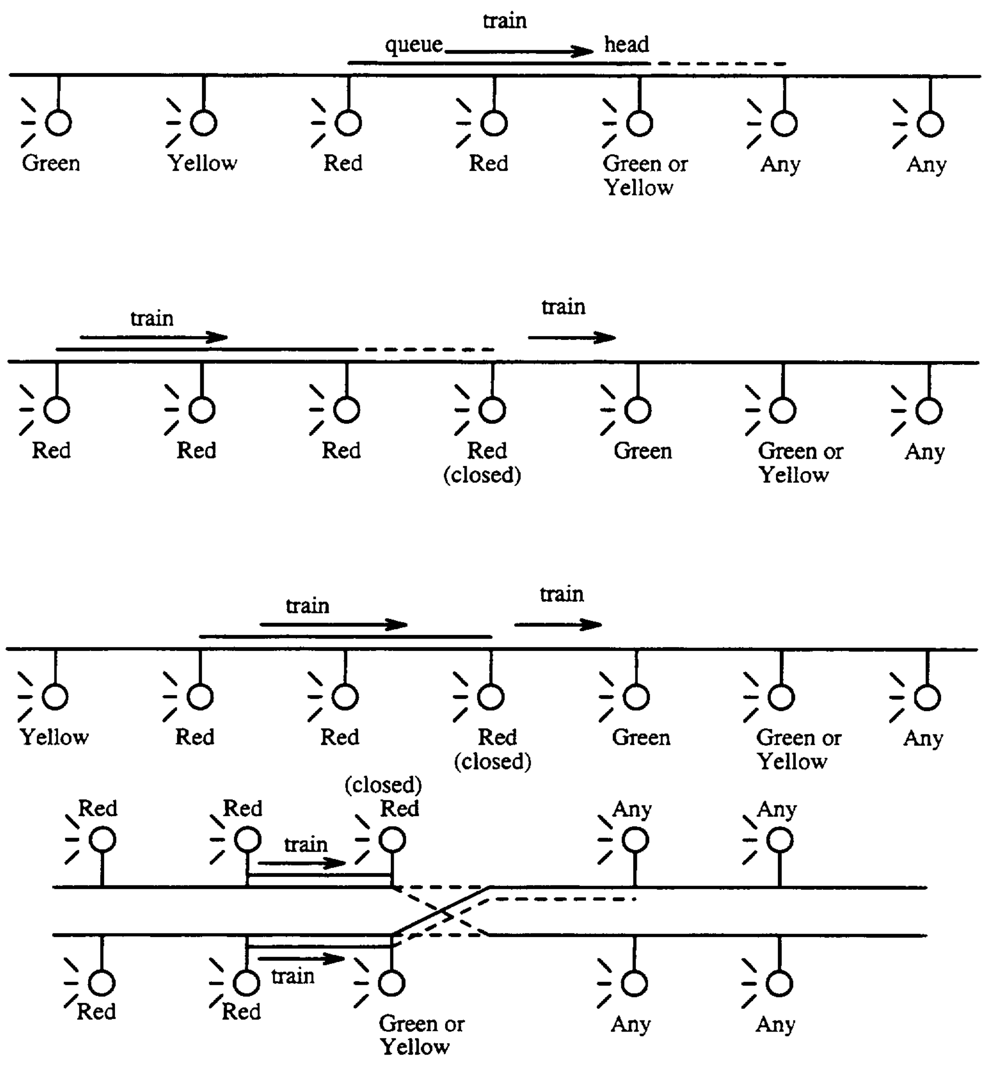
5.2.2 No block is occupied by more than one train.

5.2.3 For each train τ, if the signal s governing τ is closed, then safe_area(τ) consists only of all edges occupied by τ.

The following condition shows that condition 5.2.3 makes sense.

Lemma 5.3 *In any state satisfying condition 5.2.1, every train is governed by a signal.*

Proof. (See Figure 7.) Consider the head node v of a train τ. Either the head edge e of the train is a block, in which case there is a signal $s_{e,v}$, or else the e and v belong to a switch, and the other



Extra solid line indicates edges that must be in safe area.
Extra dashed line indicates edges that may be in safe area.

Figure 8: Safe areas of trains in various situations.

edge to which v is incident is a block B. Since the state is safe, the sector C of v, which is also the sector of B, is oriented compatibly with the position of τ. That is, v must be an entry node of B. If v' is the corresponding exit node, the signal $s_{B,v'}$ exists and governs τ.

5.7 Safe Areas and the Refinement Mapping

Theorem 5.4 *If S is a safe state of the block model, then, for each train τ, safe_area(τ) is*

5.4.1 a subpath of the netstate,

5.4.2 compatible with all sector states,

5.4.3 does not contain any edge occupied another train.

Proof. Let S be a safe state and let τ be a train.

Proof of 5.4.1.

By 5.2.1, all edges in the position of τ belong to the netstate. Any additional blocks occupied by τ must belong to the netstate, because blocks always do. By Lemma 5.3, the signal s governing τ exists. The signal s' following s can exist only if all edges on the path from s forward to s' exist and belong to the netstate.

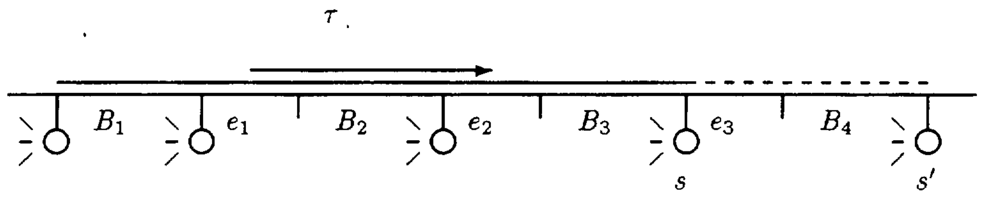
Figure 9 illustrates why the edges of *safe_area*(τ) actually form a path.

- Necessarily, *position*(τ) is oriented compatibly with the component of the signal s governing τ.
- Any additional blocks occupied by τ are adjacent to edges in *position*(τ). The flow condition implies that any such blocks must be orientable compatibly with *position*(τ).
- There is no gap between the head edge of τ and the signal S governing τ because if the head edge of τ is not actually the block B governed by s, then the head edge of τ is adjacent to B, and hence B is occupied by τ.
- If the signal s' following s exists, the path from s forward to s' is a subpath of the component of s and is contiguous with the rest of the safe area of τ.

Hence the whole safe area of τ forms a subpath of the netstate.

Proof of 5.4.2.

By 5.2.1, *position*(τ) is compatible with all sector states. Any additional block B occupied by τ shares a vertex with *position*(τ); hence compatibility of *position*(τ) with the state of sector of B implies that *safe_area*(τ) orients B compatibly with the state of the sector of B. If the path from the signal s governing τ forward to the signal following s belong to *safe_area*(τ), then 5.2.3 and 5.1.3 imply that the component of s, and hence its subpath *safe_area*(τ), is compatible with the sector of the block governed by s'. The safe area of τ does not share a vertex with a block of any sectors other than these, so it is compatible with the states of all sectors.



B_1, B_2, B_3 are blocks, e_1, e_2, e_3 are other edges.

Position of τ consists of e_1, B_2, e_2.

B_1 and B_3 are additional occupied blocks.

Signal s governs τ.

Path from s to s' consists of e_3, B_4.

Figure 9: Why the safe area of a train is a path.

Proof of 5.4.3.

By 5.2.2 no other train occupies any edge of the position of τ or any additional block occupied by τ. By 5.2.3 and 5.1.3, if the path from the signal s governing τ forward to the signal following s belong to *safe_area*(τ), then they are not occupied by any train. □

Theorem 5.4 and Lemma 4.3 now give us the following Corollary.

Corollary 5.5 *The map*

$$\rho : (S_{\text{net}}, S_{\text{signal}}, S_{\text{train}}, S_{\text{sector}}) \mapsto (S_{\text{net}}, S_{\text{train}}, S_{\text{sector}})$$

maps safe states of block model to safe states of the sector model.

Hence, in any safe state S, distinct trains have disjoint safe areas, and the map

$$\rho' : (S_{\text{net}}, S_{\text{signal}}, S_{\text{train}}, S_{\text{sector}}) \mapsto (S_{\text{net}}, S_{\text{train}})$$

maps block-safe states to basic-safe states.

5.8 State Transitions in the Block Control System

Definition 5.6 *A state transition $S \vdash_{\text{block}} S'$ is permitted in the block control model if and only if the following conditions hold.*

5.6.1 $S_{\text{net}} \vdash_{\text{net}} S'_{\text{net}}$.

5.6.2 *For any switch T, $S_{\text{net}} \cap E_T = S'_{\text{net}} \cap E_T$ (the switch's state does not change) unless every signal at a node of T is closed in S and in S' and no edge of T is occupied by a train in S.*

5.6.3 *For each train τ, one of the following two conditions holds.*

- *position$'(\tau)$ reverses position(τ) and safe_area$'(\tau)$ consists of all edges occupied by τ in S'. In this case, any sectors sharing a node with position(τ) must also reverse.*
- *position$'(\tau)$ advances position(τ), moving the head forward by at most one edge, and only within its safe area.*

  If the signal s governing τ in S does not govern τ in S' (τ has passed s), then safe_area$'(\tau)$ is as follows.

  (i) *If* aspect(s) = red *then* safe_area$'(\tau)$ *consists of all edges occupied by τ in S'.*

  (ii) *If* aspect$(s) \neq$ red *then* safe_area$'(\tau)$ *consists of all edges occupied by τ in S' plus all edges forward to the signal s'' following the signal s' governing τ in S'.*

  If the same signal s governs τ in both S and S', then safe_area$'(\tau)$ *is as follows.*

  (iii) safe_area$'(\tau)$ *includes all edges occupied by τ in S'.*

  (iv) *If* safe_area(τ) *includes edges as far the signal following the signal s governing τ in S and S', then so does* safe_area$'(\tau)$.

  (v) *If* aspect(s) = red *and* safe_area(τ) *includes only edges occupied by τ in S, then* safe_area$'(\tau)$ *contains only edges occupied by τ in S'.*

  (vi) *Otherwise,* safe_area$'(\tau)$ *may (but need not) include edges forward to the signal following s, but no others.*

5.6.4 *A signal s open in state S is open in state S' unless one of the following conditions holds.*

- *In S, the edges governed by s and the signal preceding it are unoccupied.*
- *In S', the block following s is occupied (train passing a signal).*

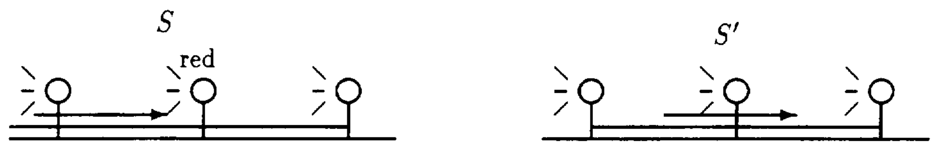
5.6.5 *A sector C may change state only if*

- *every signal s such that C is the sector following s, s is closed in S and in S' and*
- *every train whose S-position contains a node of C reverses in S'.*

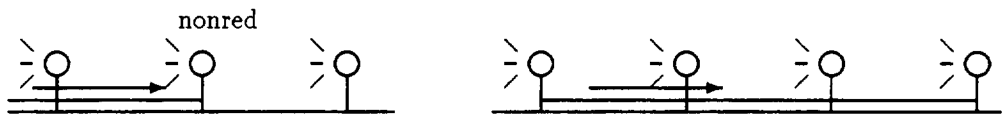
(Of course, the restrictions on states given in 5.1 must also hold in S'.)

Let us explain the state transition rule concerning safe areas.

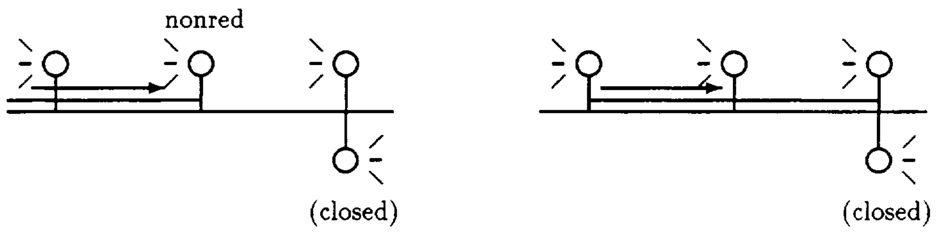
- If a train passes a non-red signal, there is nothing forcing it to stop before the next signal, so we also add the block following that signal to its safe area.



Advancing past a red signal.



Advancing past a nonred signal.



Governing signal nonred, safe area may extend.

Figure 10: How the safe area changes when a train advances.

- If the signal governing a train is not red, then as soon as the train comes in sight of that signal, the next block should be added to the train's safe area. The idea of a train sighting the signal governing is represented only by this state transition; it is not represented explicitly.
- If the safe area of a train includes the block following the signal s governing the train, and the signal s subsequently turns red or goes dark, the train may be moving too fast to stop before reaching the signal. Hence the block following s remains in the safe area. This requirement conforms to the basic model, which requires that a state change advance the safe area of each train that does not reverse.

It is trivial that if S is safe and $S \vdash_{\text{block}} S'$ then the S'-position of each train is contained in the train's S-safe area. It is also easy to prove another one of the requirements for ρ' to be a refinement mapping.

Lemma 5.7 *If S is a safe state of the block model, $S \vdash_{\text{block}} S'$, and τ is a train such that position$'(\tau)$ advances position(τ), then* safe_area$'(\tau)$ *advances* safe_area(τ).

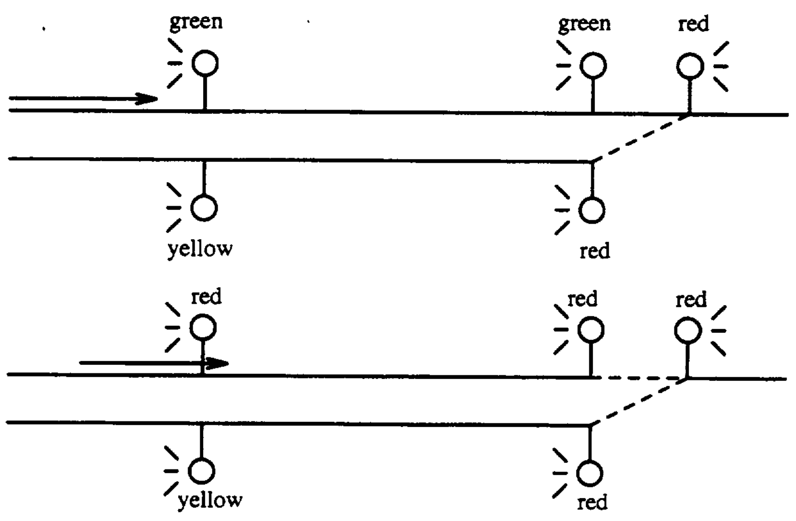

green
green
red
yellow
red
red
red
red
yellow
red
A switch opens just as a train enters an adjacent block.
The train expects to cross the switch, and will not be able
to stop in time.

Figure 11: Why two vacant blocks are required for a signal to close..

Proof. Trivial. □

5.9 Two Examples

Figures 11 and 12 illustrate the reasons for two features of the block model.

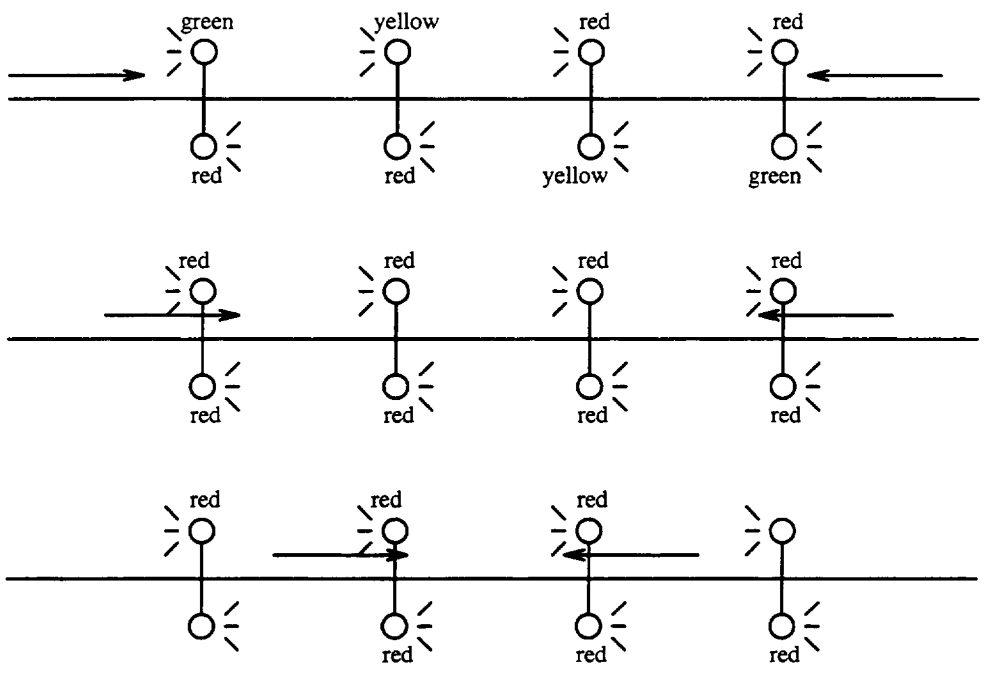
Figure 11 shows why we require *two* vacant blocks preceding a signal before we allow it to close and the switch it guards to open.

Figure 12 shows why it is necessary to include the sector model in the block model: otherwise two or even three blocks between trains is not enough to prevent head-on collisions.

5.10 The Safety Theorem

If $S \vdash_{\text{block}} S'$ is a state transition and s governs τ in S but not in S', and *position*$'(\tau)$ advances *position*(τ), then we say that τ *advances past* s in going from S to S'.

Lemma 5.8 *If S is a safe state and $S \vdash_{\text{block}} S'$ and a signal s is closed in S, then no train advances past s from S to S'.*



Two trains three blocks apart simultaneously cross block boundaries.
By the time they see the red signal, it is too late to stop.

Figure 12: Why all trains on a sector must be moving in the same direction.

Proof. If a closed signal s governs a train τ, then the safe area of the train consists only of the edges it occupies, which all precede s.

Theorem 5.9 *If S is a safe state of the block model and $S \vdash_{\text{block}} S'$ then S' is a safe state of the block model.*

Proof. We must prove that S' satisfies each of the defining conditions of safety as given in Definition 5.2.

Proof of 5.2.1.

Consider a train τ. Arguing as in Lemma 5.7, no switch crossed by *safe_area*(τ) can change state, so *safe_area*(τ) is contained in S'_{net}. Since *position*$'(\tau)$ is contained in *safe_area*(τ), it is also contained in S'_{net}.

Now consider compatibility. If the state change reverses τ, then all sectors containing vertices of *position*(τ) must also reverse, maintaining compatibility. Suppose, then, that *position*$'(\tau)$ change advances *position*(τ). Let C be a sector with which *position*$'(\tau)$ shares a node and let s be the signal governing τ in S. One of the following must hold.

- C shares a node with *position*(τ).
- The signal s is open and C contains the block following s.

In the first case, C does not reverse because it shares a node with a train that does not reverse. In the second case, C may not reverse because s is open. In either case, the S'-state of C is the same as the S-state of C, which is compatible with *position*(τ) and hence with the latter's subpath *position*$'(\tau)$.

Proof of 5.2.2.

In S', distinct trains do not occupy the same block, because all blocks they occupy are contained in the S-safe areas of the trains, which also share no edges.

Proof of 5.2.3.

We need only consider the three circumstances in which *safe_area*$'(\tau)$ can extend beyond the signal s' governing τ in S'.

- The signal s' governs τ in S and *safe_area*(τ) extends past s'. Since S is safe, s' is open in S and the block B'' following s', since it belongs to *safe_area*(τ), does not belong to the S-safe area of any other train. Hence B'' is not occupied in S or in S'. Since s' governs τ and is open in S and the block following s' exists and is unoccupied in S', s' must remain open in S'.
- The signal s' governs τ in S but *safe_area*(τ) does not extend past s'. Then s' was not red in S, hence was open in S. Hence the sector of block B'' following s' was oriented compatibly with the component of s'. Hence it can belong to the S-safe area of some train ahead of τ only if occupied in S, which it is not, since s' is not red. Hence B'' is not occupied in S'. Hence, as in the preceding case, s' is open in S'.
- τ advances past a signal s, which is not red. Then s', which is the signal following s in both S and S', is open in S, and arguing as before, remains open in S'.

This completes the proof. □

Theorem 5.9 and Lemma 4.3 now yield our final result.

Corollary 5.10 *The mapping ρ is a refinement of the generic model into the block model.*

5.11 Adapting the Block Model to Prove Properties Related to Safety

We have developed and proved safety of a particular form of the block model designed to capture the notion of safety in its strongest form: trains will never collide or cross unsafe track, not even in emergency situations, no matter how negligent their engineers may be, as long as signaling and emergency brake triggering mechanisms work as required and certain rules for setting switches are followed.

Our block model is, however, quite flexible and can be adapted to show other desirable properties of block control. For example, it is undesirable for emergency braking, which will be triggered whenever a train passes a red signal, to occur on a regular basis. Hence we would like to show that under normal operation, that is, assuming that any train that passes a yellow signal will come to a stop before passing the next signal, then in fact no train will ever pass a red signal. The nontrivial requirement is to show that the signal ahead of a train cannot be red unless the last signal passed by the train was yellow.

To model this idea, we change the safety and state change conditions that involve safe areas to the following.

- In a safe state, if the signal s governing a train τ is red, then the safe area of τ consists only of all edges occupied by τ.
- Suppose a train τ advances past a signal s. If the aspect of s before being passed was green, then in the new state S' the safe area of τ extends to the signal following the signal governing τ in S'. Otherwise, in S' the safe area of τ consists only of the edges occupied by τ.
- If the signal s governing a train τ is green or yellow, then the safe area of τ may be extended to the signal following s.
- A signal may not turn red if it or the signal preceding it governs a train.

These conditions can be obtained from the conditions of our regular definition by making the replacement

$$\begin{gathered}\textit{closed} \mapsto \textit{red}\\ \textit{red} \mapsto \textit{yellow}\\ \textit{yellow} \mapsto \textit{green}\end{gathered}$$

Note that the last of our new rules imply a change in conditions in which a switch may be opened: a signal may close (and the switch it guards subsequently open) only if the three blocks preceding it are unoccupied.

The strong condition about when a signal can turn red is needed to ensure that a train can approach a red signal only if it has previously passed a yellow signal. If we change the state transition rules to forbid that in the same transition a train τ pass a signal s and the signal s' following s turn red (that is, τ passes s just as it would have turned yellow), then we can weaken the condition on signals turning red to the following.

- A signal may not turn red if it governs a train.

What does safety of that model prove? That if engineers observe yellow lights, then emergency braking will not occur unless a train passes a signal just as it would have turned yellow.

6 Conclusion

We have presented a basic model of railway safety based on the concept of the safe area of a train, a region implicitly reserved for it by the control system and refinements of this model that respectively include control of direction of traffic on track sectors that do not contain switches and classical railway control based on blocks and signals. We have demonstrated the flexibility of this model by showing how it can be modified in order to show several safety-related properties of block control. Our basic model can also be refined so as to support safety proofs for other control systems, but doing so is beyond the scope of this paper.

The ease with which we were able to modify our block model in order to prove different safety properties suggest that it may be possible to prove a parameterized form of the safety theorem from which particular safety proofs could be obtained by plugging in appropriate parameters. Such a proof might also prove safety of systems with shorter blocks (trains can stop within n blocks instead of within one block) and more complex signaling that indicates occupancy of several blocks ahead.

We mentioned that a large part of our block control model has been formalized and lemmas proved about it in the PVS theorem prover. Finishing the formalization would be useful in order to facilitate experimentation with modified control systems and their safety proofs. Other ways to continue this work would include liveness proofs for the block model, that is, giving reasonable conditions under which state change is possible and, more generally, movement of trains toward their destination is possible; and modeling a communication protocol for sector direction control and for train control involving communication between trains and controllers.

We hope that our proof will help identify the essential concepts in railway safety and facilitate the discovery and proof of safety of new railway control protocols.

References

[1] John H. Armstrong. *The Railroad—What It Is, What It Does*. Simmons-Boardman, 1978.

[2] The Electrical Journal. *Railway Signaling*, 1908.

[3] Kirsten Mark Hansen. Validation of a railway interlocking model. In Naftalin et al. [5], pages 582–601.

[4] Trevor King. Formalising British Rail's signalling rules. In Naftalin et al. [5], pages 45–54.

[5] Maurice Naftalin, Tim Denvir, and Miquel Bertran, editors. *FME '94: Industrial Benefit of Formal Methods*, volume 873 of *Lecture Notes in Computer Science*, Barcelona, Spain, October 1994. Springer-Verlag.

[6] Revue Générale des Chemins de Fer, June 1990. No. 6.

[7] SACEM, general description.

[8] N. Shankar, S. Owre, and J. M. Rushby. A tutorial on specification and verification using PVS. Technical report, SRI International, Menlo Park CA 94025 USA, March 1993.

[9] Andrew Simpson. A formal specification of an automatic train protection system. In Naftalin et al. [5], pages 602–617.

[10] U.S. Congress Office of Technology Assessment. *Automatic Train Control in Rail Rapid Transit*, May 1976.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1996	3. REPORT TYPE AND DATES COVERED Contractor Report

4. TITLE AND SUBTITLE A Mathematical Model for Railway Control Systems	5. FUNDING NUMBERS WU 505-64-50-03 NAS1-20335
6. AUTHOR(S) D. N. Hoover	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Odyssey Research Associates, Inc. 301 Dates Dr. Ithaca, NY 14850-1326	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Langley Research Center Hampton, VA 23681-0001	10. SPONSORING / MONITORING AGENCY REPORT NUMBER NASA CR-198353

11. SUPPLEMENTARY NOTES

Langley Technical Monitor: James L. Caldwell
Final Report

12a. DISTRIBUTION / AVAILABILITY STATEMENT Unclassified - unlimited Subject Category 59	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

We present a general method for modeling safety aspects of railway control systems. Using our modeling method, one can progressively refine an abstract railway safety model, sucessively adding layers of detail about how a real system actually operates, while maintaining a safety property that refines the original abstract safety property. This method supports a top-down approach to specification of railway control systems and to proof of a variety of safety-related properties.

We demonstrate our method by proving safety of the classical block control system.

14. SUBJECT TERMS Formal Methods, Railroad Safety, Formal Modeling, and Life-Critical Systems	15. NUMBER OF PAGES 40
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT