Automated Urban Train Control with Hybrid Event-B: 'Tackling' the Rugby Club Problem

Richard Banach[a]

[a]*Department of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, U.K.*

Abstract

Normally, the passengers on urban rail systems remain fairly stationary, allowing for a relatively straightforward approach to controlling the dynamics of the system, based on the total rest mass of the train and passengers. However, when a mischievous rugby club board an empty train and then run and jump-stop during the braking process, they can disrupt the automatic mechanisms for aligning train and platform doors. This is the Rugby Club Problem for automated urban train control. A simple scenario of this kind is modelled in Hybrid Event-B, and sufficient conditions are derived for the prevention of the overshoot caused by the jump-stop. The challenges of making the model more realistic are discussed, and a strategy for dealing with the Rugby Club Problem, when it cannot be prevented, is contemplated.

1. Introduction

With profuse apologies to Clement Clarke Moore: *'Twas early in the morning, when all thro' the house, Not a creature was stirring, not even a mouse ...* aside, that is, from the stout adherents of a rugby club, who were bent on making their way to the *Métro* station, to board the otherwise empty first service of the day on the fully automated, unmanned line.[1]

As the train pulls out of the station, the dynamical variables are measured by the train system,[2] in order to gauge the weight of the passengers that have got on board — this, in order to be able to accurately predict the braking force that will be needed when the train pulls into the next station. The train becomes cognisant of the weight of the rugby club, at this point standing at the back of the train.

As the train starts to approach the next station, the rugby club start a run up the empty train towards the front. The velocity feedback control law governing the train's travel detects a shortfall in velocity and commands additional acceleration to bring the train up to speed, thereby adding to the momentum of the whole train. The train starts to brake as it enters the next station. As the train is coming to a stop, the rugby club complete their run with a jump-stop, impulsively imparting their momentum to the train body. The train has calculated its braking force on the basis of the earlier, stationary rugby club, and has not taken into account the additional momentum. As a result of the jump-stop, the train's braking force is inadequate, and the train overshoots its intended stopping point ... by a sufficient distance for the misalignment with the platform side doors to exceed the permitted safety margin. The only option for such excess misalignment (taking into account the demands of rush-hour throughput) is that the train does not stop but continues to the next station. Having given a cheer, the rugby club make their way to the back of the train, which still works on the basis of the original weight estimate. As the next station

Email address: richard.banach@manchester.ac.uk (Richard Banach)

[1] Such as the Paris *Météor* Line 14, engineered using the B-Method.

[2] Acceleration, time taken to reach cruising speed, etc.

is approached, they start a run ... you can guess the rest. On a circular line,[3] the rugby club can amuse themselves this way all day long, with the train never stopping until the end of service. This is the Rugby Club Problem for automated urban railways.[4]

The problem of a moderate, but nevertheless unacceptable overshoot of the door position by an automated urban train is easily solved if the train doors are equidistantly spaced. Then, it is enough to have an additional door or two at the front end of the platform. The train then aims for its normal position, and if an overshoot happens, the train can carefully, but quite quickly, inch along to the next spare door position, the equidistant spacing guaranteeing that all train doors will thereby be correctly aligned.[5] But the equidistant design is not widely adopted. To have enough doors per carriage to cope with a busy rush hour in an urban environment that is populated enough to justify an urban rail solution in the first place, puts considerable demands on the structural integrity of the carriages, leading to additional costs.[6]

Putting aside levity, the aim of this paper is to demonstrate that Hybrid Event-B [20, 21] can deal fluently with the problem of modeling the kind of impulsive physics exhibited by the Rugby Club Problem. By now, there are quite a number of published case studies using Hybrid Event-B [19, 18, 12, 16, 23, 15, 14], but none of the ones published hitherto has focused on impulsive physics to the extent that the present case study does.

The remaining sections of the paper are as follows. In Section 2, we outline Hybrid Event-B, emphasising the elements that are useful in modelling impulsive physics. In Section 3, we introduce an initial model of the Rugby Club Problem, focusing on just the epochs of the case study which are captured in the values of relevant mode variables, but ignoring the detailed continuous dynamics for the time being. In Section 4, we formally refine the preceding model by including the continuous dynamics, which now brings the impulsive elements clearly to the fore. This demonstrates how the issues raised by impulsive physics are handled in Hybrid Event-B, and we reflect on how these issues impact the formal system associated with Hybrid Event-B.

In Section 5, we consider various other factors that can affect the dynamics of train braking, and overview alternatives and enhancements that could be added to the model. The following two sections examine these in more detail. Section 6 discusses how the model may incorporate additional deterministic resistive forces in the dynamics. Section 7 considers the possibility that some aspects of the resistive dynamics might be subject to uncertainty, and discusses a model that incorporates these in a simple way. One outcome of these efforts is the realisation of how rapidly the complexity of the derivations increases as these additional features are included. Section 8 discusses how the Rugby Club Problem might be addressed in the context of the modelling of this paper and of its various enhancements. In Section 9 we cover related work. Section 10 concludes.

This paper is an extended version of the conference treatment of the same problem [17]. Compared with

[3] O. K. The *Météor* line is not circular on the Paris Métro map, but you get the idea. (Actually, the *Météor* line *is* circular behind the scenes, featuring loops at its two ends (beyond the section open to passengers) that join the tracks in the eastward and westward directions. The thoughtful reader may wonder at this. After all, it is much less costly to simply extend the tunnel containing the bidirectional track open to passengers with further straight sections at either end containing points to allow crossover and change of direction (assuming that the trains are symmetrical under reflection, which is always the case these days). However, switching the points takes some time, and points are also prone to failure (which the author can attest to from his experience on the Manchester Metrolink system), both of which threaten the Paris rush hour throughput and dependability requirements.)

[4] This delightful story originates with Thiérry Lecomte [55], who describes it as an outcome of a safety brainstorming session at ClearSy [29]. Although evidently a little fanciful from a real world perspective, it was seized on by the author as providing a valuable workout for the capabilities of the Hybrid Event-B formalism.

[5] Such a design is visible on the Shanghai Metro's circular line 4, as well as on some other, older Shanghai Metro lines, built when train door alignment control was less precise than today.

[6] The robustness of the carriages on the Shanghai line 4 would put much heavy rail to shame.

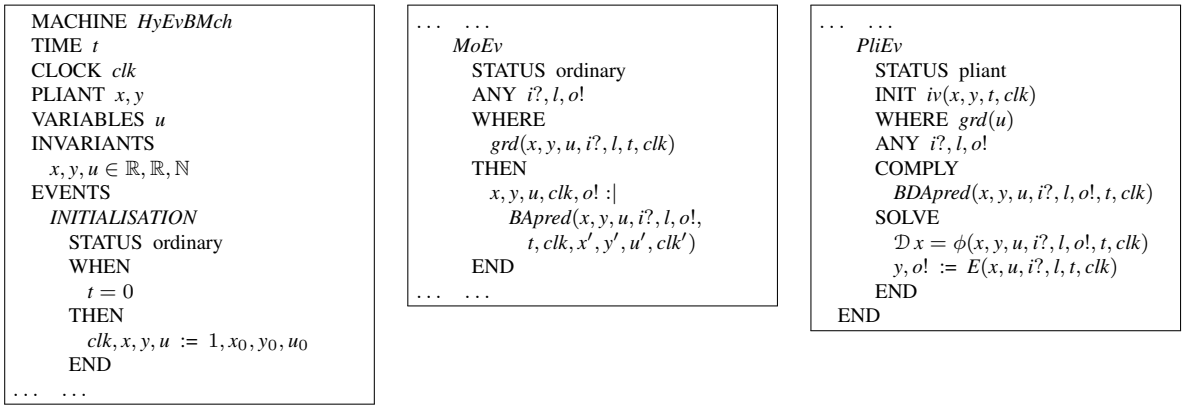

Figure 1: A schematic Hybrid Event-B machine.

[17], which was hampered by lack of space, in this paper we give a more detailed coverage of Hybrid Event-B and of methodological considerations. Also, we have used the additional space available to make the modelling more detailed (in particular by building the principal model up via refinement from the mode level version, thus exemplifying the main methodological paradigm we discuss), and including additional appropriate material. In the later sections of the paper, we have explored extensions (and their consequences) of the original simple deterministic modelling approach, illustrating the additional complexity that rapidly accrues when one does so. There is also a section on related work. These parts are new.

2. Hybrid Event-B, and Modelling Impulsive Physics

In this section we outline Hybrid Event-B for a single machine, which specifies behaviour in a single thread of control; full details are contained in [20]. Concurrent behaviour is handled via the cooperation of several machines; we do not need that aspect in this paper, for which see [21]. To set the scene, we highlight the main distinction between Hybrid Event-B and its predecessor, Event-B. An Event-B machine specifies a set of sequences of state changes. Insofar as these are intended to model behaviour of a real world system, they are normally understood to take place at distinct moments of real world time. Hybrid Event-B axiomatises this separation in real time of instantaneously executed discrete state transitions, and interleaves them with episodes of continuously varying behaviour, capable of describing the behaviour of physical and engineering systems.

In Fig. 1 we see a bare bones Hybrid Event-B machine, *HyEvBMch*. It starts with declarations of time and of a clock. In Hybrid Event-B time is a first class citizen in that all variables are functions of time, whether explicitly or implicitly. However time is special, being read-only and never being assigned, since time cannot be controlled by any human-designed engineering process. Clocks allow a bit more flexibility, since they are assumed to increase their value at the same rate that time does (i.e. one unit per unit of time), but may be set during mode events (see below).

Variables are of two kinds. There are mode variables (like u, declared in the usual manner) which take their values in discrete sets and change their values via discontinuous assignment in mode events. There are also pliant variables (such as x, y), declared in the PLIANT clause, which take their values in topologically dense sets (normally $\mathbb{R}$) and which are allowed to change continuously; these changes are specified via pliant events (see below).

Next are the invariants. These resemble invariants in discrete Event-B, in that the types of the variables are asserted to be the (static) sets from which the variables' values *at any given moment of time* are drawn. More complex invariants are similarly predicates involving all the variables that are required to hold *at all moments of time* during a run.

Then we get to the events. The *INITIALISATION* has a guard that synchronises time with the start of any run (the WHEN clause), while all other variables are assigned their initial values in the usual way (in the THEN clause that complements the WHEN clause). As hinted above, in Hybrid Event-B, there are two kinds of event: mode events and pliant events.

Mode events are direct analogues of events in discrete Event-B. They can assign all machine variables (except time itself). In the schematic *MoEv* of Fig. 1, we see three parameters $i?, l, o!$, (an input, a local parameter, and an output respectively), and a guard *grd* which can depend on all the machine variables, and defines mode event enabledness. We also see the generic after-value assignment specified by the before-after predicate *BApred*, which can specify how the after-values of all variables (except time, inputs and locals) are to be determined. The usual abbreviations using assignment notation such as := are available.

Pliant events are new to Hybrid Event-B. They specify the continuous evolution of the pliant variables over an interval of time. The schematic pliant event *PliEv* of Fig. 1 shows the structure. There are two guards: there is *iv*, for specifying enabling conditions on the pliant variables, clocks, and time; and there is *grd*, for specifying enabling conditions on the mode variables. Their conjunction defines pliant event enabledness. The separation between the two guards is motivated by considerations connected with refinement (discussed in detail in [20]).

The body of a pliant event contains three parameters $i?, l, o!$, (once more an input, a local parameter, and an output respectively) which are functions of time, defined over the duration of the pliant event. The behaviour of the event is defined by the COMPLY and SOLVE clauses. The SOLVE clause specifies behaviour fairly directly using two specification mechanisms: direct assignments and ordinary differential equations (ODEs). For example the behaviour of pliant variable y and output variable $o!$ is given by a direct assignment to the (time dependent) value of the (vector valued) expression $y, o! := E(\ldots)$. By contrast, the behaviour of pliant variable x is given by the solution to the first order ODE $\mathcal{D}\, x = \phi(\ldots)$, where $\mathcal{D}$ indicates differentiation with respect to time. (In fact the semantics of the $y, o! := E$ case can be given in terms of the ODE $\mathcal{D}\, y, \mathcal{D}\, o! = \mathcal{D}\, E$, so that x, y and $o!$ satisfy the same regularity properties.) The COMPLY clause can be used to express any additional constraints that are required to hold during the pliant event via its before-during-and-after predicate *BDApred*. Typically, constraints on the permitted range of values for the pliant variables, and similar restrictions, can be placed here.

The COMPLY clause has another purpose. When specifying at an abstract level, we do not necessarily want to be concerned with all the details of the dynamics — it is often sufficient to require some global constraints to hold which express the needed safety properties of the machine's plaint events. (Often these are refined to more deterministic behaviour at lower levels of abstraction.) The COMPLY clauses of the relevant pliant events can house such constraints directly, leaving it to lower level refinements to add the necessary details of the dynamics.

If, from Fig. 1, we erase time, clocks, pliant variables and pliant events, we arrive at a skeleton (conventional) Event-B machine. This simple erasure process illustrates (in reverse) the way that Hybrid Event-B has been designed as a clean extension of the original Event-B framework. The only difference of note is that, now —at least according to the (conventional) way that Event-B is interpreted in the physical world— (the mode) events (left behind by the erasure) execute *lazily*, i.e. *not* at the instant they become enabled (which is, of course, the moment of execution of the previous event).

2.1. Semantics of Single Hybrid Event-B Machines

This section summarises the essentials of single machine Hybrid Event-B semantics that we need for the models of this paper. This is taken from [20], where further details and references can be found.

For a machine, such as *HyEvBMch*, the semantics is an operational semantics that constructs *system traces*. A system trace is a set of functions of time, one for each variable v declared in *HyEvBMch*,

recording the value of v throughout a run of the machine. The semantics $\mathcal{S}$ of *HyEvBMch*, is the set of all system traces.

Time is modeled as an interval $\mathcal{T}$ of the reals. A run starts at some initial moment of time, t_0 say, and lasts either for a finite time, or indefinitely. So for *HyEvBMch*, $\mathcal{S}$ would consist of all system traces for clk, x, y, u, each defined over the duration of its run, which all start at $t = 0$.

Every system trace in the semantics must consist of *piecewise absolutely continuous* functions of time, with each piece being absolutely continuous on a left-closed right-open time interval such as $[t_i \ldots t_{i+1})$ where $t_i < t_{i+1}$. This is regardless of whether the piece arises from: (a) a COMPLY clause (in which case only piecewise absolutely continuous functions satisfying *BDApred* are considered); (b) an ODE with RHS which is Lipschitz continuous in the variables and measurable in time (in which case absolute continuity of the solution is guaranteed); (c) a direct assigment with RHS which is itself absolutely continuous; (d) any consistent combination of (a)-(c); (e) a mode variable's value during the interval (which remains constant except at mode transitions).

The duration of the run $\mathcal{T}$, thus breaks up into a succession of left-closed right-open subintervals: $\mathcal{T} = [t_0 \ldots t_1), [t_1 \ldots t_2), [t_2 \ldots t_3), \ldots$, in which mode transitions, effecting discontinuous updates, take place at the isolated times corresponding to the common endpoints of these subintervals t_i, and in between, the mode variables are constant and the pliant events stipulate continuous change in the pliant variables.

The operational semantics of Hybrid Event-B constructs system traces via an abstract (i.e. non-executable) algorithmic process, extending a system-trace-to-be, event execution by event execution, and replicating system-traces-to-be over available choices when choice points are encountered during an extension step. Evidently, without further control, such an approach can easily run into inconsistency. The full description in [20] contains many 'runtime checks' that pick up such inconsistencies and eliminate the corresponding system-trace-to-be from the semantics. We omit these here, firstly for brevity (since we never need them in the models below), and secondly because they can be prevented by verifying the Hybrid Event-B proof obligations for a given model during a static analysis.

The construction of system traces for a machine M can be summarised as follows.

[1] Let $\eta := 0$.

[2] CHOOSE an initial assignment to all variables satisfying all the invariants of M, thereby interpreting their values at time t_0.

[3] With the state variables having the values at t_η, CHOOSE an enabled pliant event *PliEv* and CHOOSE a simultaneous piecewise absolutely continuous solution, in a maximal left-closed, right-open interval $[t_\eta \ldots t_{\text{MAX}})$, of all the differential equations and direct assignments in the SOLVE clause of *PliEv*, using state variable values at t_η as initial values, with these initial values required to satisfy the INIT and WHERE guards of *PliEv*, and with inputs and local parameters where needed, such that *BDApred* in the COMPLY clause of *PliEv* is also satisfied in the interval, and all the invariants of M are maintained. Use the solution to assign the values of all pliant variables (and outputs) in $[t_\eta \ldots t_{\text{MAX}})$.

[3.1] For every mode variable, extend its value at t_η to a constant function in the interval $[t_\eta \ldots t_{\text{MAX}})$.

[4] If no non-*INITIALISATION* mode event is enabled by the values of the state variables at any time in the open interval $(t_\eta \ldots t_{\text{MAX}})$ (including left-limit at t_{MAX} itself), together with a choice of values for inputs and local parameters, then TERMINATE.

[5] CHOOSE $t_{\eta+1} > t_\eta$ such that **either** $t_{\eta+1}$ is the earliest time at which a non-*INITIALISATION* mode event without inputs is enabled, **or** a non-*INITIALISATION* mode event with inputs is enabled at $t_{\eta+1}$ and there is no non-*INITIALISATION* mode event without inputs that is enabled within $(t_\eta \ldots t_{\eta+1})$.

[6] Let $\eta := \eta + 1$.

[7] CHOOSE a mode event that is enabled by the values of variables at t_η (or their left-limit values if $t_\eta = t_{\text{MAX}}$), and any needed inputs and locals, and assign to all state variables and outputs according to its *BApred*, such that all the invariants of *M* are satisfied, thereby (re)interpreting those variable values at t_η.

[7.1] For any other state variable without a value at t_η, interpret its value at t_η as its left-limit at t_η.

[7.2] Discard the interpretation of all state variables in the open interval $(t_\eta \ldots t_{\text{MAX}})$.

[8] Goto **[3]**.

That the above abstract procedure does not fail during the construction of system runs can be guaranteed by confirming the numerous Hybrid Event-B proof obligations (POs), discussed in detail in [20]. These can be summarised as follows.

- Initial states are well defined (feasible).
- Feasible initial states satisfy the invariants.
- Mode events which are enabled in invariant states have well defined after-states (i.e., are feasible).
- Feasible mode events reestablish the invariants.
- Pliant events which are enabled in invariant states have time-indexed families of well defined after-states that satisfy all the clauses in the event's specification, in some left-closed right-open time interval (i.e., are feasible). Optionally, the length of the interval must reach a Zeno lower bound.
- Feasible pliant event after-state families preserve the invariants at least until a preemption point.
- The after-state of any mode event disables mode events and enables some pliant event.
- The after-state family of any non-FINAL pliant event enables some mode event (the earliest time for this defining the pliant event's preemption point).

Besides the above, there are the Lipschitz and measurability properties of any ODE RHS to be checked. These follow readily for practical problems. The Zeno check is optional since it is usually impossible to verify it without solving the entire dynamics first, whereas the static checks are intended to justify avoiding doing exactly that. The FINAL designation permitted for pliant events is intended to prevent the last condition above from producing errors for events that are designed to complete a run.

We can summarise the above picture of the semantics in a more intuitive way thus:

[A] Every enabled mode event is feasible, i.e. has an after-state, and on its completion enables a pliant event (but does not enable any mode event).[7]

[B] Every enabled pliant event is feasible, i.e. has a time-indexed family of after-states, and EITHER:

(i) During the run of the pliant event a mode event becomes enabled. It preempts the pliant event, defining its end. ORELSE

(ii) During the run of the pliant event it becomes infeasible: finite termination. ORELSE

(iii) The pliant event continues indefinitely: nontermination.

[7] If a mode event has an input, the semantics *assumes* that its value only arrives at a time strictly later than the previous mode event, ensuring part of [A] and [B] automatically. By this means we can ensure a mode event executes asynchronously — and if the only purpose of having an input would be to ensure this asynchronous scheduling, we can introduce the 'async' status as a shorthand, and omit the input altogether, as in Figs. 2 and 3.

2.2. *Single Hybrid Event-B Machine Refinement*

Hybrid Event-B machines are developed by refinement. A concrete (refining) machine is like any other machine, with two provisos. Firstly, each concrete event must declare which abstract event it refines, unless it is a 'new' mode event — 'new' pliant events must also declare a refining abstract event. Secondly, the relationship between abstract and concrete state spaces is captured in a retrieve (or gluing) relation, also referred to as the joint invariant (supported by input and/or output relations, as needed).

A concrete machine has to obey the POs above, where 'invariants' is always interpreted to include the joint invariant with its 'dangling abstract variables' existentially quantified. Additional POs govern the refinement process itself, described in detail in [20]. Summarising, we have the following.

- If, in an invariant concrete state, a refining concrete mode event is enabled, then its abstract counterpart is enabled in a corresponding abstract state (identified via the retrieve relation).
- If, in an invariant abstract state, any abstract mode event is enabled, then in a corresponding concrete state (identified via the retrieve relation), some concrete mode event (whether refining or new) is enabled.
- If, in an invariant concrete before-state connected to a corresponding abstract before-state via the retrieve relation, a refining concrete mode event makes a transition, then there is a transition of its abstract counterpart from the abstract before-state to an after-state connected via the retrieve relation to the concrete after-state.
- If, in an invariant concrete before-state connected to a corresponding abstract state via the retrieve relation, a new concrete mode event makes a transition, then its after-state is connected via the retrieve relation to the same abstract state.
- Every transition of a concrete new mode event decreases a variant function.
- If, in an invariant concrete state, a concrete pliant event is enabled, then its abstract counterpart is enabled in a corresponding abstract state (identified via the retrieve relation).
- If, in an invariant abstract state, any abstract pliant event is enabled, then in a corresponding concrete state (identified via the retrieve relation), some concrete pliant event (whether refining or new) is enabled.
- If, in an invariant concrete before-state connected to a corresponding abstract before-state via the retrieve relation, a concrete pliant event makes a transition, then there is a transition of its abstract counterpart from the abstract before-state, such that for all times during that transition, the current abstract and concrete states are connected via the retrieve relation.

The last of these is based on the premise that time flows at the same rate in abstract and concrete models. For theoretical convenience, we assume the sets of variables used in the two machines are disjoint. But for refinements which just add variables and behaviour to an existing model (as we typically do in this paper), we include the abstract variables among the concrete variables and presume the retrieve relation to be the natural projection from concrete to abstract.

With these POs verified, it becomes possible to prove that every concrete run has a simulating abstract run with corresponding transitions matching at suitable times during the run [20].

2.3. *Hybrid Event-B and the Modelling of Physics*

The mode events of Hybrid Event-B, which permit the discontinuous state changes of the computational world to be represented, also allow impulsive physics to be conveniently modelled. For example, a billiard cue strikes a ball, changing its velocity discontinuously, or a capacitor discharges, instantaneously reducing the electrical potential across its plates to zero. However, unlike the computational world in which the programmer is at liberty to decide what discontinuous state changes take place, the

physical world is governed by immutable physical laws, hard won through extensive work in the laboratory, which must be adhered to to yield a useful model. Thus, in the billiard ball example, it is the conservation of momentum that determines the relationship between the physical states before and after the strike. In the capacitor example, the instantaneous discharge presumes that the other elements in the electrical circuit can react to the concomitant impulsive changes in voltage and current that the discharge generates. We might say that *'Hybrid Event-B cannot do your physics for you; but it can faithfully represent the physics that you know from elsewhere.'*

Connected with the preceding is the fact that discontinuous state changes in the physical world are stimulated by forces which are 'pure impulses'. And whereas discontinuous change can be represented quite directly in Hybrid Event-B, these pure impulses cannot. Physicists and engineers speak of such impulses as 'delta functions' — 'zero everywhere except at a single point, were they are infinite, but with a finite integral'. Mathematically, that last phrase is meaningless; delta functions are not functions, but so-called distributions [80, 77, 50]. The nearest we get in Hybrid Event-B (or any other similar formalism) to the representation of a pure impulse is the (syntactic) description of the mode event that encapsulates the discontinuous change that results from the impulse. The occurrence of the mode event (at runtime) corresponds to the occurrence of the physical impulse causing the discontinuous change of state.

3. The Rugby Club Problem Mode Level Model

In this section, we begin the presentation of a very simple model of the rugby club scenario formalised in Hybrid Event-B. Here, we describe its broad structure in terms of the epochs of the playout of the scenario, expressed via the modes that govern the behaviour. The design of Hybrid Event-B has been aimed at enabling the graduated development of hybrid systems, allowing the detailed dynamics to be incorporated later via refinement. This initial model is in Fig. 2.

The initial model itself consists of little more than the modes permitted in the models, and the events that express the transitions of the state machine whose states they embody. Thus there is only one variable *mode* at this level, and the model starts in the initial state in which *mode* describes the *STAT*ionary state of the train at the starting station. Right after the *INITIALISATION*, there is the *PliTrue* pliant event. Although we have no intention of including pliant dynamics in the model at this level, the semantics of Hybrid Event-B is fixed to be in terms of functions of physical time, modelled using the reals. Therefore, in models at the mode-only level, we include a default pliant event such as *PliTrue* to satisfy the exigencies of the semantics: it merely stipulates that the invariants be maintained any time it runs (which, in models at this level, means at all times between the runtime occurrences of the mode events (which the semantics stipulates must occur at isolated moments of time)).

In order to save some space (particularly with the later Fig. 3 in mind), we have economised a bit on the notation. Primarily, we have decanted events' STATUS declarations to a decoration at the end of the line containing the event name. In this paper, the STATUS can be 'pliant', with the obvious meaning; or the STATUS can be 'async', meaning that it is a mode event for every execution of which a strictly positive amount of time since the execution of the preceding mode event must elapse; or the STATUS can be 'ordinary' —normally not written, by analogy with Event-B— meaning that it is a mode event which must execute eagerly, i.e. as soon as its guard becomes true.

In the present model, there are only mode variables and mode events. In such a situation, since executing a mode event immediately enables its successor, the default eager semantics of Hybrid Event-B would demand that the successor event executes at the same moment of time as its predecessor. Since this is unphysical, Hybrid Event-B semantics forbids it. The modelling inconvenience of this is overcome by the feature suggested in footnote 7, i.e. we can introduce an unneeded input to the mode event. Hybrid Event-B semantics then restricts its moment of execution to be *other than* the moment of execution of

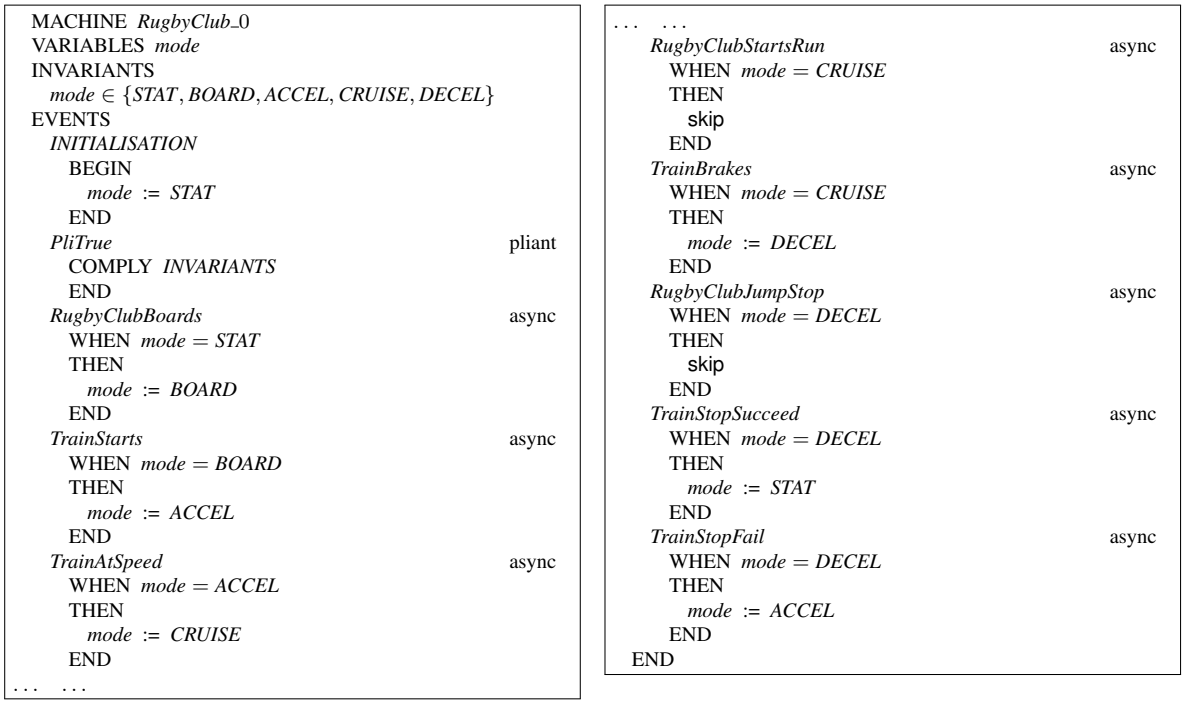

Figure 2: A mode level Hybrid Event-B model of the urban rail Rugby Club Problem.

any preceding mode event occurrence. In this manner we can make an event's moment of execution depend on an *external choice* (rather than depending purely on internal elements of the model itself, which would make it an *internal choice*).

We use the 'async' STATUS to act as a shorthand for the modelling trick just described. And since there are only mode variables and mode events in the present model, *all* mode events in the model must be async. Refinement, in permitting the acquisition of additional guards in an event, may guarantee that a successor mode event cannot be enabled immediately that its predecessor completes, obviating the need for the async status. Thus an async mode event may be refined to an ordinary one. We see some instances of this in the next section.

The *STAT*ionary epoch is brought to an end by the Rugby Club *BOARD*ing the train. The async status of the event signifies that its moment of execution is chosen by the environment and not just by the model's internal state. The *BOARD*ed epoch ends when the train starts, changing the mode to *ACCEL*erating. The *ACCEL*erating epoch ends when the train reaches cruising speed and then the mode becomes *CRUISE*.

At some point during the *CRUISE*ing epoch, the *RugbyClubStarts*their*Run*. This will change the as yet not introduced dynamical variables, but we intend our modelling to be sufficiently simple that we can assume that this happens in a fixed way that can be described using constant assignments, so that the effect of the event is in fact idempotent, and we do not need to change the *mode* variable (though, of course, we could have chosen otherwise).

At some point during the *CRUISE*ing epoch, the *TrainBrakes*, and this assigns the *mode* variable to the value *DECEL*erating. Since *mode* was not changed when the Rugby Club's run started, the model has no way of knowing if the run actually started or not. This imposes an obligation on the subsequent modelling to default to normal behaviour if the Rugby Club were in fact behaving well and not running.

While the train is *DECEL*erating, the *RugbyClubJumpStop*s. Given the uncertainty regarding whether the Rugby Club is running or not, this must also default to normal behaviour if the Rugby Club were

not running. And since *mode* does not change during this event, the event's semantics must also be idempotent, as for *RugbyClubStartsRun*.

The two events *TrainStopSucceed* and *TrainStopFail* capture the two possible eventual outcomes of the *DECEL*erating epoch, when the train's braking strategy has either succeeded or failed. The further specifics of these events require dealing with the more detailed dynamics, in the next section.

4. The Rugby Club Problem Model Refined

In this section, we refine the mode level model of the *RugbyClub_0* machine of Fig. 2 to include the detailed dynamics. This brings to the fore the insights about impulsive physics discussed earlier. The refined model itself is in the *RugbyClub_1* machine of Fig. 3.

The more detailed dynamics of machine *RugbyClub_1* depends on a number of constants (which would be declared in a CONTEXT, which we do not show). Thus we have *BIGT*, an initial value for a clock that is bigger than any value that could trigger the enabledness of any mode event; M_T, the mass of the train; M_{rc}, the mass of the rugby club; V_{cr}, the cruising speed of the train; V_{rcr}, the rugby club's running speed relative to the train's speed (when the members of the rugby club are, in fact, running).

A number of variables contain the state of the model. Evidently there is *mode*, inherited from *RugbyClub_0*. There are a number of new variables. Some represent mass: m_{in}, the inertial mass of the system at any time; m_{pcv}, the mass perceived by the train at any time (based on the dynamical properties that it measures and the moments in time that it does so). There are also: v_{rcr}, the rugby club's running speed relative to the train at any time (regardless of whether the rugby club are, in fact, running or not at that time); *brTime*, the train's concept of the needed duration for the braking period, at the start of the braking period. These variables are all mode variables, because they only need to get updated via mode events, so only acquire a discrete number of values during any execution.

There are also pliant variables: m_{eff}, the effective mass of the system, i.e. the point mass which, when traveling at the train's velocity, would possess the same momentum as the whole train plus rugby club system, thus offering the same resistance to change in momentum as the whole system — it changes continuously when the rugby club is running during acceleration or braking, due to the continuously changing relative proportions that the train and the rugby club contribute to the overall momentum during the accelerating or braking episodes; v_T, the speed of the train at any time; *brDist*, the current remaining distance during the braking period until the train comes to a standstill, as computed by the train according to the dynamical properties that it measures.

In reality, not all of these variables are strictly necessary. Many can be dispensed with as they can be re-expressed in terms of constants and other variables. The variables in this category are: m_{in}, m_{pcv}, v_{rcr}, m_{eff} and *brTime*. We nevertheless retain them in order to make the ensuing explanation of the model easier to follow.

The invariants are trivial typing invariants in this simple model: *mode* is as described earlier, and the others are all either reals or non-negative reals. We discuss some possibilities for more complex invariants later.

We turn to what the model actually does. In order to save space in Fig. 3, we have introduced some further notational economies. Thus, we have slightly generalised the CONST declaration of [12] to cover a list of (pliant) variables that are to stay constant during the execution of a pliant event. Also we have omitted the REFINES declarations from the text of the events. These are easy to reinstate. Thus REFINES *PliTrue* applies to all the pliant events, and for the mode events, each of them REFINES the identically named mode event of machine *RugbyClub_0*.

INITIALISATION starts the model with the train stationary in a station with no one on board. A clock *clk_A*, is set to an innocuous value *BIGT*; the *mode* is *STAT*; all the masses are set to be the train's inertial

MACHINE *RugbyClub_1*
REFINES *RugbyClub_0*
CLOCK *clk_A*
VARIABLES $mode, m_{in}, m_{pcv}, v_{rcr}, brTime$
PLIANT $m_{\text{eff}}, v_T, brDist$
INVARIANTS
$mode \in \{STAT, BOARD, ACCEL, CRUISE, DECEL\}$
$m_{in}, m_{pcv}, v_{rcr}, brTime \in \mathbb{R}^+, \mathbb{R}^+, \mathbb{R}^+, \mathbb{R}^+$
$m_{\text{eff}}, v_T, brDist \in \mathbb{R}, \mathbb{R}, \mathbb{R}$
EVENTS
INITIALISATION
BEGIN
$clk_A := BIGT$
$mode := STAT$
$m_{in}, m_{pcv} := M_T, M_T$
$v_{rcr} := 0$
$m_{\text{eff}} := M_T$
$v_T := 0$
$brDist := 0$
$brTime := 0$
END
TrainStationary pliant
WHEN $mode = STAT \vee mode = BOARD$
THEN
COMPLY CONST$(m_{\text{eff}}, v_T, brDist)$
END
RugbyClubBoards async
WHEN $mode = STAT$
THEN
$mode := BOARD$
$m_{in} := M_T + M_{rc}$
$m_{\text{eff}} := M_T + M_{rc}$
END
TrainStarts async
WHEN $mode = BOARD$
THEN
$mode := ACCEL$
$clk_A := 0$
END
TrainAccelerating pliant
WHEN $mode = ACCEL$
COMPLY CONST(m_{eff})
THEN
$\mathcal{D}\, v_T = F_A / m_{in}$
$brDist := 0$
END
TrainAtSpeed
WHEN $mode = ACCEL \wedge v_T = V_{cr}$
THEN
$mode := CRUISE$
$m_{pcv} := (F_A\, clk_A)/V_{cr}$ // $\ldots = m_{in} = (M_T + M_{rc})$
$m_{\text{eff}} := m_{in}$
END
TrainCruising pliant
WHEN $mode = CRUISE$
THEN
$\mathcal{D}\, v_T = -K(v_T - V_{cr})$
$brDist := 0$
$m_{\text{eff}} := m_{pcv} + \dfrac{M_{rc} v_{rcr}}{v_T}$
END
... ...

... ...
RugbyClubStartsRun async
WHEN $mode = CRUISE$
THEN
$v_{rcr} := V_{rcr}$
$v_T := V_{cr} - \dfrac{M_{rc}\, V_{rcr}}{M_T + M_{rc}}$
$m_{\text{eff}} := \dfrac{(M_T + M_{rc})}{\left[1 - \dfrac{V_{rcr}}{V_{cr}}\left[1 + \dfrac{M_T}{M_{rc}}\right]^{-1}\right]}$
END
TrainBrakes async
WHEN $mode = CRUISE \wedge |v_T - V_{cr}| < \epsilon$
THEN
$mode := DECEL$
$brDist := -\dfrac{m_{pcv}}{F_D}\dfrac{V_{cr}^2}{2}$
$brTime := \dfrac{m_{pcv} V_{cr}}{F_D}$
END
TrainDecelerating pliant
WHEN $mode = DECEL$
THEN
$\mathcal{D}\, v_T = -\dfrac{F_D}{m_{pcv}}$
$\mathcal{D}\, brDist = v_T$
$m_{\text{eff}} := m_{pcv} + \dfrac{M_{rc} v_{rcr}}{v_T}$
END
RugbyClubJumpStop async
WHEN $mode = DECEL \wedge v_T > 0$
THEN
$v_{rcr} := 0$
$v_T := \dfrac{m_{\text{eff}}}{m_{in}} v_T$
$m_{\text{eff}} := m_{pcv}$
END
TrainStopSucceed
WHEN $mode = DECEL \wedge v_T = 0 \wedge$
$|brDist| \leq BRTOL$
THEN
$clk_A := BIGT$
$mode := STAT$
$m_{in}, m_{pcv} := M_T, M_T$
$brDist := 0$
$brTime := 0$
END
TrainStopFail
WHEN $mode = DECEL \wedge v_T = 0 \wedge$
$|brDist| > BRTOL$
THEN
$clk_A := 0$
$mode := ACCEL$
$brDist := 0$
$brTime := 0$
END
END

Figure 3: A simple Hybrid Event-B model of the urban rail Rugby Club Problem.

mass M_T; the train's velocity and the rugby club's relative velocity are set to zero; and all other variables are of no interest and are also set to zero.

The ensuing pliant event *TrainStationary* just perpetuates this state of affairs — all mode variables cannot change, and the pliant variables are held constant via the CONST declaration.

At some point during this phase the async event *RugbyClubBoards* is executed. Although boarding clearly does not take place instantaneously, only the overall change in mass makes any difference, and so there is no harm in modelling the process as an impulsive change to the mass during this event. The inertial mass m_{in} becomes $M_T + M_{rc}$, as does the effective mass m_{eff} (since the train system behaves as a single mass at this stage). Everything else stays the same. In particular, the train's perceived mass m_{pcv} remains unchanged since the train is, as yet, unaware of the rugby club. After this the *TrainStationary* event resumes, all variables maintaining their values, whether old values or newly acquired values.

At some point after this the dynamics starts, and for this, we assume an conventionally idealised setup. Thus we assume the track is straight and level, the movement of the train is frictionless and suffers no other impediment (such as air resistance), and the train can be treated as one (or several) point mass(es) for the purpose of dynamical calculations.

In complex situations, dynamics is best treated via the d'Alembert-Lagrange approach, or an equivalent. See e.g. [39, 34]. The foundations are not in fact as uncomplicated as the ancient pedigree of this subject might suggest; [25] gives a good discussion, not to mention the gargantuan [65]. For us, it will suffice to stick to a fairly low-level approach, *provided* we remember that Newton's Second Law equates *force* to rate of change of *momentum*, and not to mass times acceleration, as is usually stated, and to which the more accurate form usually reduces.

So, async event *TrainStarts* executes. It changes the mode to *ACCEL*erating and starts the clock. It thus enables the *TrainAccelerating* pliant event which states how the pliant variables change. Since the rugby club are stationary, the effective mass m_{eff} remains CONSTant at its value at the start of *TrainAccelerating*. The braking distance variable *brDist* is not needed yet, so is kept at zero.[8] The nontrivial element of the *TrainAccelerating* event is the ODE that equates the rate of change of the train's momentum $\mathcal{D}\left(m_{in}\, v_T\right)$ to the force applied by the train. The latter is assumed to be a constant accelerating force F_A. Since there is no relative motion between the train and rugby club, and all the train and rugby club mass is treated as concentrated at the centre of gravity, we can take the mass element to be the inertial mass m_{in}, and we derive the statement found in Fig. 3.

Acceleration continues until the train achieves cruising velocity, detected by the guard $v_T = V_{cr}$ of the mode event *TrainAtSpeed*. The moment that this happens —observe that this is a case in which an async event has been refined to an ordinary one, permissible because of the stronger guard it has acquired, which prevents it being enabled at the moment the previous mode event executed— this event turns off the accelerating force and changes the mode to *CRUISE*ing. This also enables the train to calculate its overall perceived mass m_{pcv} from the information it has, namely clock value clk_A, applied force F_A and cruise velocity V_{cr}. Of course, since the motion has been so simple thus far, a straightforward application of Newtonian mechanics (namely, that change of momentum $m_{pcv} \times V_{cr}$ equals duration of applied force $F_A \times clk_A$) shows that the answer m_{pcv}, is m_{in}, as noted in the accompanying comment, but the train can only use the information available to it, so we show the assignment to m_{pcv} expressed using only those quantities.

[8] This could also have been handled via a CONST declaration. In fact, that would have been more convenient, since assignment to a (time dependent, in general) expression generates a verification condition to check that the initial value of the expression agrees with the value on entry to the pliant event, in order to ensure right continuity of the variable's history at the entry point to the pliant event, as required by the semantics [20]. Not mentioning *brDist* at all would entail the default behaviour for pliant variables during pliant events, namely of constraining them to simply obey any relevant invariants. This would be inappropriate here.

TrainAtSpeed enables the *TrainCruising* pliant event. *brDist* is still not needed, so is assigned as previously. The train velocity v_T is controlled by a linear constant coefficients ODE, impelling v_T towards the stable equilibrium value V_{cr}. Since $v_T = V_{cr}$ immediately after *TrainAtSpeed*, there is no change in velocity at this time. Similarly, the effective mass m_{eff} remains as before, which is easy to see in the direct assignment to m_{eff} when we notice that $v_{rcr} = 0$ during this period.

While *TrainCruising* runs, async mode event *RugbyClubStartsRun* is enabled, and at some point is executed. Now the dynamics gets more interesting. Again we idealise the change of state as an impulsive change, since only the overall change in momentum matters, and the dynamics is completely lossless. The rugby club's relative velocity with respect to the train v_{rcr}, becomes V_{rcr}. Since momentum is conserved, using primes for after-values, we can write:

$$(M_T + M_{rc})v_T = (M_T + M_{rc})v'_T + M_{rc}V_{rcr} \tag{1}$$

from which, noting that $v_T = V_{cr}$, we derive the assignment to the after-value v'_T that we see in *RugbyClubStartsRun*. The train effective mass m_{eff} becomes the mass that is needed to generate the momentum on either side of (1) when the velocity is the new train velocity. A slightly longer calculation, equating (1) to $m'_{\text{eff}}v'_T$, is needed to derive the expression for the after-value m'_{eff} (given in terms of the cruise velocity V_{cr}) that appears in *RugbyClubStartsRun*. Note that for simplicity we have made *RugbyClubStartsRun* idempotent; it does not change *mode* and thus does not disable itself, and its assignments are to constant expressions. So several occurrences of it could take place, with occurrences after the first amounting to skip, and the end effect on the dynamical variables would be the same as for one occurrence.

After *RugbyClubStartsRun*, *TrainCruising* is still enabled. Since the train velocity is no longer V_{cr}, the feedback control law in *TrainCruising* now has work to do. Implicitly, an accelerating force is applied to impel the train velocity v_T towards V_{cr}, and it does work that adds to the total momentum of the system. Note that as v_{rcr} is non-zero, having become V_{rcr}, and given that v_T changes, so does m_{eff}, as can be derived from (1), reflecting the changing proportion of the overall momentum that the rugby club's relative run velocity contributes.

When v_T has returned close enough to V_{cr}, the async mode event *TrainBrakes* becomes enabled — we are assuming that the train velocity has recovered before the train starts to brake. We assume the train knows where it is relative to the next stopping position, and initiates braking at a point where, according to the train's perception about its dynamics, applying its fixed braking force F_D for an appropriate time will bring it to a halt just where needed. We assume that the train still imagines its overall mass is the originally calculated m_{pcv}, and taking the velocity to be V_{cr}, a simple Newtonian mechanics calculation of the (quadratic) displacement generated by a constant force yields the *brDist* value assigned in *TrainBrakes*, assuming further that the next stopping position is the origin of distance measurements, and that positive distances are oriented beyond the stopping position. The time taken to come to a halt is recorded in *brTime* — it is just the time needed to consume all of the assumed momentum $m_{pcv}V_{cr}$ by applying a force of magnitude F_D.

TrainBrakes changes the mode to *DECEL*erating, and thus immediately enables the behaviour in the pliant event *TrainDecelerating*. During this period, it is the laws of physics, and not the train's perceptions, that determine what happens. Thus, the rate of change of velocity is governed by the momentum form of Newton's Law:

$$\mathcal{D}\left((M_T + M_{rc})v_T + M_{rc}V_{rcr}\right) = -F_D \tag{2}$$

In (2), only v_T can vary, the other symbols being constants. Thus by rearranging (2), we derive the ODE for v_T that appears in *TrainDecelerating*. And v_T gives the time derivative of *brDist*. The effective mass m_{eff} is given by the same formula as in *TrainCruising*, for exactly the same reasons.

At some point during *TrainDecelerating*, but while the velocity is still positive, the rugby club come to the end of their run. The momentum that they 'stole' from the train when they initiated their run, and which was unknowingly made up by the feedback control law during *TrainCruising*, is now suddenly dumped back into the train when they do their jump-stop.

The physical consequences of this process are described in the *RugbyClubJumpStop* async mode event. The rugby club relative run velocity v_{rcr} changes from V_{rcr} to zero. Since the train system now behaves once more like a point mass, the effective mass must likewise become m_{in}. The process is governed by conservation of momentum, which, using primes for after-values as usual, yields the following:

$$m_{\text{eff}} v_T = m_{in} v'_T = m_{pcv} v'_T = m'_{\text{eff}} v'_T \tag{3}$$

This explains the assignments to the variables in *RugbyClubJumpStop*. Note that *RugbyClubJumpStop* is another idempotent mode event, though it is idempotent for more complicated reasons than previously. Thus, it does not disable itself, and although the assignment $v_T := \frac{m_{\text{eff}}}{m_{in}} v_T$ is not to a constant expression (whereas the other assignments *are* to constant expressions), the accompanying assignment $m_{\text{eff}} := m_{pcv}$ ensures that any repeated execution of $v_T := \frac{m_{\text{eff}}}{m_{in}} v_T$ effects no change in v_T.

Once *RugbyClubJumpStop* has executed, *TrainDecelerating* is enabled once more. But now, the train velocity which the decelerating phase has to deal with is suddenly greater than before. So the braking phase is necessarily extended compared with its previously anticipated duration (although the train is not aware of this).

It is intuitively clear that if the rugby club consists of extreme lightweights, and that if they run extremely slowly, the effect on the braking episode will be small due to the small amount of momentum at issue. Equally, if the rugby club are all very heavy, and they run very fast, then the effect on the braking episode will be more appreciable.

Mode events *TrainStopSucceed* and *TrainStopFail* handle these two possibilities. One or other is triggered when v_T hits zero (and the mode is still *DECEL*erating). These are two more mode events which have been refined from async ones, acquiring stronger guards and thus becoming ordinary. The ideal stopping position is at $brDist = 0$. So if the discrepancy between the ideal and actual stopping positions when v_T hits zero does not exceed *BR*aking*TOL*erance, then *TrainStopSucceed* executes, and the train stops at the station, as it should. The state returns to its initial configuration assuming that the rugby club have alighted (presumably disappointed). The whole scenario can then repeat.

However, if the discrepancy between the ideal and actual stopping positions when v_T hits zero exceeds *BR*aking*TOL*erance, then *TrainStopFail* executes, the train returns to *ACCEL*erating mode, and the train moves towards the next station, with the (presumably elated) rugby club on board. In this case we have the classic Rugby Club Problem scenario which can then also repeat.

Note, by the way, that *TrainStopSucceed* also covers the possibilities that *RugbyClubStartsRun* and/or *RugbyClubJumpStop* never executed at all — after all, their execution is merely *enabled* (and not *forced*, as would be the case if no other event were enabled). In the case that *RugbyClubStartsRun* did not execute, then: the effective mass m_{eff} does not get changed, *RugbyClubJumpStop* (if it executed) would become equivalent to skip, and *TrainStopSucceed* would just implement the normal dynamics. In the case that *RugbyClubJumpStop* did not execute, then: if *RugbyClubStartsRun* had executed, the momentum stealing would have taken place but no momentum restitution would have happened, and the rugby club's additional momentum would have remained decoupled from the momentum of the train. So, as the train comes to a stop at the station in the normal manner, the rugby club presumably leap off it on the run, to conform to the conditions of our modelling.

4.1. Analysis of the Jump-Stop Phenomenon

In this section we analyse the distinction between the *TrainStopSucceed* and *TrainStopFail* cases more precisely.

During an execution of *TrainDecelerating*, if the intial velocity of the train is v_{IN} and the pliant event executes for a time t_{EX}, then after this period, the velocity and distance travelled become:

$$v_{IN} - \frac{F_D\, t_{EX}}{(M_T + M_{rc})} \qquad \text{and} \qquad v_{IN}\, t_{EX} - \frac{F_D\, t_{EX}^2}{2(M_T + M_{rc})} \tag{4}$$

respectively. To work out the implications of the jump-stop, we need to consider two such episodes, separated by a *RugbyClubJumpStop*.

The braking period starts with the train moving forward with velocity V_{cr}. Suppose the rugby club do their jump-stop t_{JS} later than the start of braking. Then, substituting into (4), after the first braking episode, the velocity and distance travelled become:

$$v_{JS} \;=\; V_{cr} - \frac{F_D\, t_{JS}}{(M_T + M_{rc})} \qquad \text{and} \qquad d_{JS} \;=\; V_{cr}\, t_{JS} - \frac{F_D\, t_{JS}^2}{2(M_T + M_{rc})} \tag{5}$$

Then comes the jump-stop. According to *RugbyClubJumpStop*, the velocity needs to be rescaled by m_{eff}/m_{in}, which increases the velocity expression in (5) to:

$$v'_{JS} \;=\; v_{JS} \left[m_{in} + \frac{M_{rc}\, V_{rcr}}{v_{JS}} \right] \Big/\, m_{in} \;=\; v_{JS} + V_{rcr} \left[1 + \frac{M_T}{M_{rc}} \right]^{-1} \tag{6}$$

$$\;=\; V_{cr} - \frac{F_D\, t_{JS}}{(M_T + M_{rc})} + V_{rcr} \left[1 + \frac{M_T}{M_{rc}} \right]^{-1} \tag{7}$$

Braking is then completed by another *TrainDecelerating* episode. This time the initial velocity is v'_{JS}. Using (4) with this initial value, the pliant behaviour executes until the velocity drops to zero. Naming this duration t_{HALT}, it is given by:

$$v'_{JS} - \frac{F_D\, t_{HALT}}{(M_T + M_{rc})} \;=\; 0 \qquad \text{thus} \qquad t_{HALT} \;=\; \frac{(M_T + M_{rc})\, v'_{JS}}{F_D} \tag{8}$$

and therefore, the distance covered in the second *TrainDecelerating* episode is, by (4):

$$d_{HALT} \;=\; v'_{JS}\, t_{HALT} - \frac{F_D\, t_{HALT}^2}{2(M_T + M_{rc})} \tag{9}$$

The total distance travelled during braking is therefore $d_{TOT} = d_{JS} + d_{HALT}$, subject to the constraint that $v_{JS} > 0$. If we call $brDist_{TOT}$ the (negative) value assigned by *TrainBrakes* to the variable *brDist*, it is the discrepancy between d_{TOT} and $brDist_{TOT}$ that must be compared to *BRTOL* to determine whether *TrainStopSucceed* or *TrainStopFail* will be enabled. We find:

$$\begin{aligned}
d_{TOT} \;=\; & V_{cr}\, t_{JS} - \frac{F_D\, t_{JS}^2}{2(M_T + M_{rc})} + v'_{JS}\, t_{HALT} - \frac{F_D\, t_{HALT}^2}{2(M_T + M_{rc})} \\
\;=\; & V_{cr}\, t_{JS} - \frac{F_D\, t_{JS}^2}{2(M_T + M_{rc})} + \frac{(M_T + M_{rc})\, v'^2_{JS}}{F_D} - \frac{F_D\, (\frac{(M_T + M_{rc})\, v'_{JS}}{F_D})^2}{2(M_T + M_{rc})} \\
\;=\; & V_{cr}\, t_{JS} - \frac{F_D\, t_{JS}^2}{2(M_T + M_{rc})} + \frac{(M_T + M_{rc})\, v'^2_{JS}}{2\, F_D} \\
\;=\; & V_{cr}\, t_{JS} - \frac{F_D\, t_{JS}^2}{2(M_T + M_{rc})} \\
& + \frac{(M_T + M_{rc})}{2\, F_D} \left(V_{cr} - \frac{F_D\, t_{JS}}{(M_T + M_{rc})} + V_{rcr} \left[1 + \frac{M_T}{M_{rc}} \right]^{-1} \right)^2
\end{aligned} \tag{10}$$

We expand the quadratic term, which causes some cancellations. After some working we get:

$$d_{TOT} = \frac{(M_T + M_{rc})}{2\,F_D}\left[V_{cr}^2 + V_{rcr}^2\left[1+\frac{M_T}{M_{rc}}\right]^{-2} + 2\,V_{rcr}V_{cr}\left[1+\frac{M_T}{M_{rc}}\right]^{-1} - 2\,V_{rcr}\frac{F_D\,t_{JS}}{(M_T+M_{rc})}\left[1+\frac{M_T}{M_{rc}}\right]^{-1}\right] \tag{11}$$

So

$$d_{TOT} < \frac{(M_T + M_{rc})}{2\,F_D}\left[V_{cr}^2 + V_{rcr}^2\left[1+\frac{M_T}{M_{rc}}\right]^{-2} + 2\,V_{rcr}V_{cr}\left[1+\frac{M_T}{M_{rc}}\right]^{-1}\right] \tag{12}$$

The last step follows, because in the last two terms of (11), the negative one cannot exceed the positive one in magnitude. This follows because in (5) the problem requirements force v_{JS} to be positive, whence t_{JS} is bounded above. When t_{JS} takes its maximum value, v_{JS} is 0, and it is then easy to check that when this holds, the last two terms of (11) cancel.

To surmount the Rugby Club Problem, it is sufficient to arrange that *TrainStopSucceed* always executes at the end of the braking process and *TrainStopFail* never does. For this, it is sufficient that the value of *brDist* when $v_T = 0$ (let us call this quantity *brDist*0, noting that it is equal to $brDist_{TOT} + d_{TOT}$), satisfies $|\,brDist0\,| < BRTOL$.

We can use this insight to create an additional, nontrivial state invariant (13), where below, *HYP* denotes the relationships between the various constants of the model that have to be true in order that the required condition $|\,brDist0\,| < BRTOL$ can be proved:

$$HYP \;\vdash\; mode = DECEL \wedge v_T = 0 \;\Rightarrow\; |\,brDist\,| \leq BRTOL \tag{13}$$

Thus, (13) states that the assumptions guarantee that at the crucial moment, the *brDist* variable has reached a value within the margin *BRTOL* of zero, and so the enabledness of *TrainStopSucceed* at the crucial moment becomes provable (and, corrrespondingly, that the non-enabledness of *TrainStopFail* at the crucial moment is also provable).

Regarding $brDist_{TOT} + d_{TOT} = brDist0$, which it is easy to see equals the last two terms of (12), we note that the V_{rcr}^2 term will be negligible in magnitude compared with the $V_{rcr}V_{cr}$ term. This enables us to derive a simple criterion that will be adequate for most engineering purposes:

$$brDist0 \approx \frac{V_{rcr}\,V_{cr}\,M_{rc}}{F_D} \tag{14}$$

From this we derive a simple and practically adequate condition to reassure us that the Rugby Club Problem isn't a problem:

$$V_{rcr}V_{cr}M_{rc}/F_D \leq BRTOL \tag{15}$$

4.2. Formal Issues Connected with the New Invariant

It is instructive, at this point, to consider how the preceding observations would be reflected in a treatment of this scenario that was mechanically supported by a Rodin-like tool equipped with the capabilities of dealing with the requirements of the relevant applied mathematics. We sketch this in the present section.

Regarding (13), first of all, the contents of *HYP* would be held in a suitable CONTEXT, that the *RugbyClub*_1 machine SEES. The part of (13) after the turnstile would constitute the clause that would

be added to the INVARIANTS. As with all invariants, it would be necessary to show that it is established by the *INITIALISATION*, and that all events maintain it. We review this now. Since *brDist* is initialised to 0, it is immediate that (13) is established at the start, and is maintained by all events in the left hand column of Fig. 3, and by *RugbyClubStartsRun* too.

TrainBrakes also maintains it, since its guard insists that the train velocity v_T is close to V_{cr}, which is away from zero, which falsifies the hypothesis of the invariant, and since v_T is not reassigned within the event, the hypothesis of the invariant remain falsified in the after-state. *RugbyClubJUmpStop* also maintains (13) since v_T is assumed non-zero and cannot become zero in the relevant assignment. (For the latter, we would need to add another invariant stating that m_{eff} can never be zero — but it is not too hard to show that this is established at the start and is maintained by all events, although the latter depends on a similar additional invariant stating that m_{pcv} is also never zero.)

The remainder is trickier. The only pliant event during whose execution the hypotheses of (13) *might* be satisfied, is *TrainDecelerating*. To show that in fact this is never possible, it is enough to show that v_T is never zero during any execution of the event.[9] To show *that*, it is sufficient to show that the value of v_T is non-zero upon entry to the event, because, based on such an assumption, as the event executes, at any moment when $v_T = 0$ might be reached, *TrainDecelerating* is preempted by one of *TrainStopSucceed* or *TrainStopFail*. To show that the value of v_T is non-zero upon entry to *TrainDecelerating*, we examine the events that enable it. These are *RugbyClubJumpStop* and *TrainBrakes*. But we have already argued that in the case of both events, v_T is non-zero in both the before-state and the after-state of the event. Therefore, since v_T is never zero during any execution of *TrainDecelerating*, we conclude that the invariant is maintained by this event.

There remain *TrainStopSucceed* and *TrainStopFail*. In the case of *TrainStopSucceed*, the invariant is assumed true in the before-state and is established by the assignment to *brDist* in the after-state, thus discharging the maintenance of the invariant. In the case of *TrainStopFail*, the invariant is assumed false in the before-state and is established by the assignment to *brDist* in the after-state, again discharging the maintenance of the invariant. So the invariant is indeed maintained, as required.

Some observations are in order at this juncture. The first is that in the above discussion, we seem to have avoided the very heavy calculations of Section 4.1. The reason is that in Section 4.1 we were concerned with the *truth* of (13), whereas above, we were merely interested in its *maintenance*. The reason that the maintenance task was easier is that the model contains the event *TrainStopFail*. This caters for the possibility that the facts needed in *HYP* for establishing the invariant do not hold. To caricature the situation, above, we were concerned with a property P, whereas the model caters for $P \vee \neg P$.

We can give more bite to the situation by assuming that the facts needed for *HYP* indeed hold, and then removing the event *TrainStopFail*. In such a case, proving the maintenance of the invariant amounts to removing the discussion of *TrainStopFail* above, actually simplifying matters. But trouble arises elsewhere. In the discussion of semantics in Section 2.1, one of the POs mentioned demanded that '*The after-state family of [a...] pliant event enables some mode event [at] the pliant event's preemption point*'. As a consequence of this (and disregarding irrelevant events), the after-state of *TrainDecelerating*[10] implies the disjunction of the guards of *TrainStopSucceed* and *TrainStopFail*, which (again disregarding irrelevant detail) amounts to $mode = DECEL \wedge v_T = 0$, i.e. the value of *brDist* is unconstrained (the

[9] In fact, if v_T could be zero on entry to *TrainDecelerating*, *TrainStopSucceed* or *TrainStopFail* would also be enabled immediately after the completion of the mode event that enables *TrainDecelerating*, violating the PO that insists on the strict alternation of executing pliant and mode events during a run.

[10] Strictly speaking, since *TrainDecelerating* executes for a time interval which is right open, there is no definitive after-state. It is the limit of the family of states in the execution of *TrainDecelerating* at the preemption time that we are referring to as the after-state.

$P \vee \neg P$ effect). Without the *TrainStopFail* event though, the disjunction amounts to $mode = DECEL \wedge v_T = 0 \wedge \mid brDist \mid \leq BRTOL$. Now, we are forced to actually consider the value of *brDist* at the preemption of *TrainDecelerating* so that we can establish the guard of *TrainStopSucceed*, bringing in the complexity we saw in Section 4.1.

This brings us to our second observation. The reader will have noticed how the considerations of Section 4.1 frequently required forwards reasoning about the effect of an event execution on the execution of the successor event, and on the execution of the successor to the successor, etc., in the manner of a simulation. This was reflected (to a lesser degree, for the $P \vee \neg P$ reasons just discussed) in backwards reasoning in our account of the maintenance of (13). Either exigency rather flies in the face of the reasoning typical of (discrete) Event-B and similar formalisms, which rely purely on reasoning about individual states. Typically, there are two approaches to this issue.

On the one hand, we may consider introducing sufficiently many additional invariants that the dynamic properties of the state trajectories are, in effect, captured within sufficiently many sufficiently detailed implications between state properties, that the facts required can be inferred. This works best in the discrete case as individual state changes are associated with individual event occurrences. Complicated situations can require the introduction of additional variables to capture information that is lost by the original variables of the model, and the number of additional invariants that need to be introduced is not guaranteed to be small. In the hybrid case, a continuum of states is normally visited during any pliant event execution. The kind of reasoning we needed to do above, showed that details of temporal behaviour are often needed. Some of these can be codified as generic properties of the class of mathematical functions being used.[11] In other cases though, time itself, instrumented appropriately (e.g. by the use of suitable clocks) may need to be included within the properties reasoned about, because of the level of detail needed.[12]

On the other hand, we can take inspiration from the style of our argument about (13), and propagate information along successions of event executions. In (discrete) Event-B event enabledness governs how this can take place, while in Hybrid Event-B, this is further constrained by the 'well formedness' POs that demand that mode events only enable pliant events, and *vice versa*. These principles are the analogues of the sequential composition rules in more conventional languages. In Hybrid Event-B, every time one event enables another, properties derived for the after-state of the former may be suitably incorporated as a potential strengthening of the guard of the latter. This is what we did above, in effect. And, of course, once we have modified some event, we can repeat the process of propagation to elicit further consequences, iterating the process until a fixed point is obtained. By doing this we would have strayed a long way into model checking territory, made flesh in the case of discrete Event-B in the ProB tool [70, 57].

5. The Rugby Club Problem — Further Discussion

The details of the control strategy actually used for urban rail control are commercially confidential, for obvious reasons. Nevertheless, it seems clear that the fact that one could imagine that there could be a rugby club problem at all, signals a likely cause of it as being the discrepancy between a control strategy based on pure kinematics and one based on the complete dynamics. In this section, we briefly discuss some issues for more realistic modelling that this realisation prompts.

[11] Detailed issues of this kind are being explored in [4], which is the successor to [20, 21], and is concerned with reasoning techniques for Hybrid Event-B.

[12] This raises interesting issues. The very word '*invariant*' implies a lack of variability over time. So invariants that do not mention time in any form are to be preferred. But the issues we are considering may force the use of invariants which are logically independent of time, but which have the form of implications whose hypotheses and conclusions are definitely *not* independent of time.

Several factors would need to be taken into account in a more realistic model: the track will not be straight and level; it will not sustain frictionless train travel; the train's wheels will not always make perfect rolling contact with the track (there will be some skidding at times); the control laws will not be as simple as we have chosen them to be in our models; in the confines of an underground tunnel, air resistance will cause significant drag on the train. And so on.

All these things will soak up some of the momentum of the train as it travels, requiring work from the engine to maintain speed. Simple realistic models of these phenomena will not necessarily be available. The best one might hope for, would be phenomenological models that predicted the relevant losses, based on tabulated data taken over many journeys under a variety of conditions. These data would have to be specific to each section of the route, and dealing with these aspects could seriously complicate the design of the critical code controlling the train's motion.

In the next couple of sections we enhance our model to capture some of what has just been mentioned, albeit in a rather elementary way. To keep matters simple, we restrict the complications to just the braking phase, the earlier part of the model remaining unchanged. We introduce some modelling of additional frictional and other resistance as an extra braking term. We also introduce uncertainty into the modelling to cover noisy and unpredictable variation in the dynamics.

6. Deterministic Resistance

6.1. Constant Resistance

The simplest way to incorporate the effect of additional resistive forces in the model is to assume that they are constant along the portion of the dynamics in which they act. If, as we said, for simplicity we restrict their effect to only act during the braking episode of the dynamics, then it is sufficient to rescale the braking force F_D to include the additional contribution: F_D becomes $F_D^R = F_D + R$ where R is the additional resistive force. Then all the conclusions of Section 4.1 remain valid with F_D^R replacing F_D. In particular, the Rugby Club Problem is not an issue provided:

$$V_{rcr} V_{cr} M_{rc} / (F_D + R) \leq BRTOL \tag{16}$$

6.2. Proportional Resistance

Unfortunately, additive models of dynamical resistance are not all that well borne out in practice. More convincing but still simple models of dynamical resistive force assume that the force is proportional to the velocity and is oppositely directed to it. This is quite well supported experimentally. A further simplifying assumption is that the relevant proportionality factor is a constant over a broad range of velocity values. Since this leads to an analytically solvable model, we will pursue this formulation further.

Suppose the constant of proportionality, rescaled by the masses, is R. This leads to the modified *TrainDecelerating* event of Fig. 4. The law for v_T in *TrainDecelerating* is a simple inhomogeneous linear ODE, analogous to the one in *TrainCruising*. Recalling that $m_{pcv} = m_{in}$,[13] if the velocity is initially v_{IN}, then after a time t_{EX}, it becomes:[14]

$$\left(v_{IN} + \frac{F_D}{R} \right) \mathrm{e}^{-\frac{R}{m_{in}} t_{EX}} - \frac{F_D}{R} \;=\; \sum_{k=0}^{\infty} \frac{(-R)^k}{k!} \left(\frac{t_{EX}}{m_{in}} \right)^k \left[v_{IN} - \frac{F_D \, t_{EX}}{(k+1)\, m_{in}} \right] \tag{17}$$

[13] We use m_{pcv} in the model to emphasise that the train can only reason about its situation on the basis of information available to it, but we use m_{in} in our discussion to emphasise that *we* are in possession of global information.

[14] The power series representation comes in useful later.

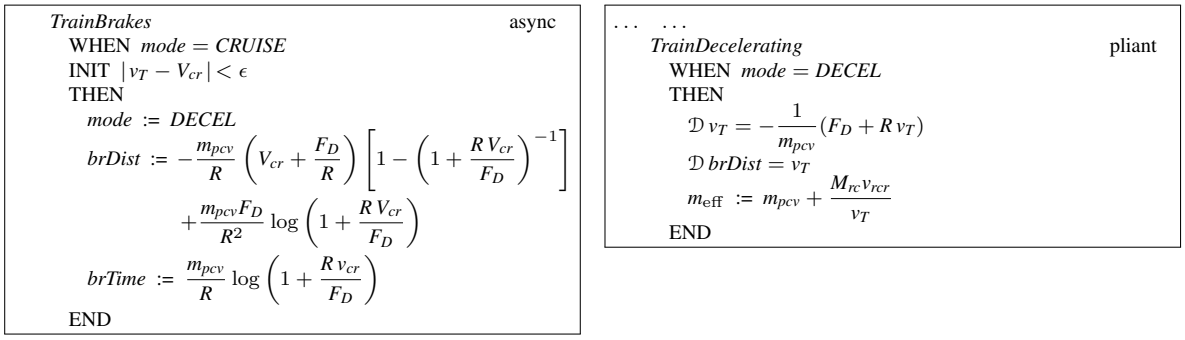

Figure 4: Adding resistive forces statically proportional to velocity to the dynamics of Fig. 3.

Integrating (17) from zero to a time t_{EX} gives the distance travelled:

$$\frac{m_{in}}{R}\left(v_{IN}+\frac{F_D}{R}\right)\left[1-\mathrm{e}^{-\frac{R}{m_{in}}t_{EX}}\right]-\frac{F_D}{R}t_{EX} \tag{18}$$

Since the train system is presumed to have been designed to be aware of the resistive forces, the calculation of the total braking time and total braking distance to be assigned in event *TrainBrakes* must be adjusted compared with the earlier case. So, absent the rugby club, the velocity becomes zero when (17) does, giving a duration of travel:

$$\frac{m_{in}}{R}\log\left(1+\frac{R\,v_{IN}}{F_D}\right) \tag{19}$$

Substituting V_{cr} for v_{IN} in (19) gives the expression assigned to *brTime* in the modified *TrainBrakes* event of Fig. 4. Now, inserting that value for t_{EX}, and V_{cr} for v_{IN}, into (18), and negating the result to account for the origin of distance being defined at the end of travel, gives the expression assigned to *brDist* in the modified *TrainBrakes* event. We can check that this accords with our expectations by checking the Taylor series for these expressions, which show that in the $R \to 0$ limit the *brDist* and *brTime* assignments reduce to the previously obtained values:

$$brDist \;:=\; -\frac{1}{2}\frac{m_{in}V_{cr}^2}{F_D}\left[1-\frac{2}{3}\frac{R\,V_{cr}}{F_D}+O((RV_{cr}/F_D)^2)\right] \tag{20}$$

$$brTime \;:=\; \frac{m_{in}V_{cr}}{F_D}\left[1-\frac{1}{2}\frac{R\,V_{cr}}{F_D}+O((RV_{cr}/F_D)^2)\right] \tag{21}$$

To gauge the effect of all this on the Rugby Club Problem, we have to retrace the calculation of Section 4.1. The instantaneous rescaling of velocity at the jump-stop remains the same, being given by (6), but now, instead of using (4) for the velocity and distance we must use (17) and (18) instead.

Now (17) and (18) are relatively tractable expressions, but working out the jump-stop dynamics using them entails rescaling (17) and using the rescaled value as the initial value in (17) again. The resulting expressions rapidly become unwieldy, so for perspicuity, we will work to first order in R and keep linear duration terms only in velocities. Reducing (17) and (18) in this manner gives:

$$v_{IN}-\frac{F_D}{m_{in}}\,t_{EX}-\frac{R\,v_{IN}}{m_{in}}\,t_{EX} \tag{22}$$

for the velocity after time t_{EX}, starting from v_{IN}, and gives:

$$v_{IN}\,t_{EX}-\frac{1}{2}\frac{F_D}{m_{in}}\,t_{EX}^2-\frac{1}{2}\frac{R\,v_{IN}}{m_{in}}\,t_{EX}^2 \tag{23}$$

for the corresponding distance travelled.

Reusing the notations of Section 4.1, we get for the first part of the jump-stop episode:

$$v_{JS} \;=\; V_{cr} - \frac{F_D}{m_{in}}\, t_{JS} - \frac{R\, V_{cr}}{m_{in}}\, t_{JS} \tag{24}$$

for the velocity, and:

$$d_{JS} = V_{cr}\, t_{JS} - \frac{1}{2}\frac{F_D}{m_{in}}\, t_{JS}^2 - \frac{1}{2}\frac{R\, V_{cr}}{m_{in}}\, t_{JS}^2 \tag{25}$$

for the distance. For the jump-stop itself, following (6), we add $V_{rcr}[1 + M_T/M_{rc}]^{-1}$ to (24):

$$v'_{JS} \;=\; V_{cr} - \frac{F_D}{m_{in}}\, t_{JS} - \frac{R\, V_{cr}}{m_{in}}\, t_{JS} + V_{rcr}\left[1 + \frac{M_T}{M_{rc}}\right]^{-1} \tag{26}$$

This now serves as the intial value v_{IN} for the second part of the jump-stop episode. Equating (22) to zero with this v_{IN} and solving for t_{HALT} to first order in R gives:

$$t_{HALT} \;=\; \frac{m\, v'_{JS}}{F_D} - \frac{R\, m\, v'^2_{JS}}{F_D^2} \tag{27}$$

This can now be used in (23) to get d_{HALT}, which, to first order in R is:

$$d_{HALT} \;=\; \frac{1}{2}\frac{m\, v'^2_{JS}}{F_D} - \frac{1}{3}\frac{R\, m\, v'^3_{JS}}{F_D^2} \tag{28}$$

As before, the total distance travelled during braking is therefore $d_{TOT} = d_{JS} + d_{HALT}$, (given by (25) and (28), with v'_{JS} given by (26)). This can be compared with *brDist*0, which to first order is:

$$\frac{1}{2}\frac{m_{in}\, V_{cr}}{F_D} - \frac{1}{3}\frac{R\, m_{in}\, V_{cr}^3}{F_D^2} \tag{29}$$

to ascertain the effect that the jump-stop will have. That these expressions have the right form can be seen by comparing them in the $R, t_{JS} \to 0, 0$ limit (and without including the jump-stop increment in velocity in d_{HALT}, obviously), where they agree.

6.3. Variable Proportional Resistance

Even if if proportional resistance is considered appropriate over a wide range of velocities, in practice it is unlikely that a single proportionality constant R would be considered credible for the entire dynamics, since conditions would be expected to vary along different parts of the route. One approach to this would be to model the 'constant' of proportional resistance as a piecewise constant function of position, and to combine instances of the work of Section 6.2 into a description of the total dynamics. Alternatively, we could consider tackling a nonconstant R that varied continuously, using a variation of parameters technique. The mounting complexity of the work in Section 6.2 (which was mostly sidestepped rather brusquely by our first order approach) dissuades us from exploring these options more deeply.

7. Stochastic Resistance

Another very reasonable presumption about the behaviour of a train in a realistic environment is to acknowledge that the motion will be subject to some uncertainty from unpredictable influences from the environment, which will affect the detailed dynamics. In this section, we develop a simple model of this, centred on the preceding work. Our approach is based on a device often used in simple models, namely the idea of considering a range of plausible values for various parameters of the model, and assessing their effect.

7.1. Uncertain Variable Resistive Dynamics

To apply the strategy just mentioned to our situation, we consider that the parameter R, controlling the magnitude of the resistive force, can vary. For this approach to be valid, it is not enough to simply consider the extremes of a plausible range of values for R, which is very tempting. If the aspects of interest in the dynamics do not vary monotonically with R, the quantities derived from the extremes of the range of values for R will give misleading answers.

Accordingly, the first task is to examine the variation of the velocity with R, so we examine the equation for this in *TrainDecelerating* in Fig. 4. It is clear from the RHS of the ODE that an increase in R, amplifies the rate of decrease of velocity, whatever the values of the other parameters of the model. This implies that an increase in R will cause a decrease in the stopping distance, which arises via the integral of the velocity.

Extremes of the ODE confirm this. When R is negligible with respect to F_D, the behaviour of v with respect to time goes as $v_{IN} - F_D\, t/m_{in}$. And when F_D is negligible with respect to R, the behaviour of v with respect to time goes as $v_{IN} \exp(-R\, t/m_{in})$. Both are nonincreasing with R. We can also check the derivative of the velocity with respect to R. Differentiating (17) gives:

$$\frac{F_D}{R^2}\left(1 - \mathrm{e}^{-\frac{R}{m_{in}}t_{EX}}\right) - \frac{t_{EX}}{m_{in}}\left(v_{IN} + \frac{F_D}{R}\right)\mathrm{e}^{-\frac{R}{m_{in}}t_{EX}} \;=\; \sum_{k=0}^{\infty}\frac{(-R)^k}{k!}\left(\frac{t_{EX}}{m_{in}}\right)^{k+1}\left[-v_{IN} + \frac{F_D\, t_{EX}}{(k+2)\, m_{in}}\right] \tag{30}$$

As in (17), the expansion in powers of R confirms that the pole terms in the R derivative cancel, as they must, and that the leading behaviour of the derivative for any starting value v_{IN}, and duration Δt which is short enough to allow R and higher powers of Δt to be neglected, is $-v_{IN}\Delta t/m_{in}$, i.e. always negative, confirming the monotonicity in R of the dynamics.

We now envisage the braking phase to be divided up into a number of episodes, indexed by a natural number $j \in 1\ldots$, each characterised by two values of R: an upper value R_j^U and a lower value R_j^L. Within each episode, we imagine that the actual value of R is uncertain, but that it is nevertheless surely confined between the relevant two values. Therefore, for each episode, the actual dynamics will be confined between the behaviours given by assuming the two extreme values for R, given the monotonicity established above. We can also assume (if necessary by splitting a single episode into two), that the jump-stop takes place at the boundary between two consecutive episodes.

Suppose now that the various episodes are delimited by values of time, known *a priori*. Thus, when the duration of the $j-1$'th episode reaches Δt_{j-1}, the $j-1$'th episode is exited, the j'th episode is entered, and the R_{j-1}^U and R_{j-1}^L values of the $j-1$'th episode are replaced by R_j^U and R_j^L in the j'th episode. We assume that the Δt_k durations are bounded below to avoid Zeno phenomena. Under these assumptions, the dynamics can be evaluated as follows.

The starting velocity of braking is assumed to be (negligibly different from) V_{cr}, which we can refer to as v_0^U and as v_0^L for convenience. At the start of the j'th episode, we have an upper velocity v_{j-1}^U and a lower velocity v_{j-1}^L. We confirm that the stopping distance is not contained in the j'th episode for either R_j^U or R_j^L (and thus for no value between them) by checking that Δt_j does not exceed the value given by (19) for either v_{j-1}^U or v_{j-1}^L and either R_j^U or R_j^L. Then, we derive the possible variation in the velocity and in the distance travelled in the j'th episode using: v_{j-1}^U and v_{j-1}^L as initial velocities, Δt_j as duration, R_j^U and R_j^L as R values, and (17) and (18). These yield d_j^U and d_j^L as possible distances travelled in the j'th episode. The possible final velocities are given by these means too, unless the end of the j'th episode witnesses the jump-stop, in which case $V_{rcr}[1 + M_T/M_{rc}]^{-1}$ has to be added, yielding v_j^U and v_j^L.

If Δt_j is long enough that both R_j^U and R_j^L produce stopping within the j'th episode, the two stopping distances can be evaluated, as earlier, and the all the individual d_k^U and d_k^L values can be accumulated to

give the total stopping distances from start of braking. If Δt_j is long enough for only R_j^U to yield braking within the j'th episode, the $j+1$'th episode (and subsequent episodes, if needed) need only consider the R^L calculation(s). The process can then be completed as already described.

It can legitimately be argued though, that delimiting the episodes by values of time is not very realistic (although the uncertainty permitted by the range of R values goes some way to alleviating that concern). More realistic perhaps is to assume that the episodes depend on position, reflecting that fact that there may be different conditions in different regions of the track, etc. In such a case the preceding outline would be modified as follows.

Suppose then that the various episodes are delimited by values of distance, known *a priori*. Thus, when the distance covered in the $j-1$'th episode reaches Δd_{j-1}, the $j-1$'th episode is exited, the j'th episode is entered, and the R_{j-1}^U and R_{j-1}^L values of the $j-1$'th episode are replaced by R_j^U and R_j^L in the j'th episode. We assume that the Δd_k distances are bounded below to avoid Zeno phenomena. Under these assumptions, the dynamics can be evaluated as follows.

The starting velocity of braking is assumed to be (negligibly different from) V_{cr}, which we can refer to as v_0^U and as v_0^L for convenience. At the start of the j'th episode, we have an upper velocity v_{j-1}^U and a lower velocity v_{j-1}^L. We calculate the upper expected braking distance $brDist_j^U$ and lower expected braking distance $brDist_j^L$ by the techniques that led to the *brDist* assignment in *TrainBrakes* in Fig. 4 using: v_{j-1}^U and v_{j-1}^L, and R_j^U and R_j^L, and (17)-(19) as appropriate. We confirm that the stopping distance is not contained in the j'th episode for either R_j^U or R_j^L (and thus for no value between them) by checking that $\Delta d_j \leq brDist_j^U$ and $\Delta d_j \leq brDist_j^L$. To calculate the exit velocities at the end of the j'th episode we must do two things, We must firstly solve $\Delta d_j = (18)$ (with v_{j-1}^U and v_{j-1}^L as velocities, and R_j^U and R_j^L as R values, in (18)) to obtain Δt_j^U and Δt_j^L, which are upper and lower values for the time taken to cover distance Δd_j. Secondly, we derive the possible variation in the velocity in the j'th episode using: v_{j-1}^U and v_{j-1}^L as initial velocities, Δt_j^U and Δt_j^L as durations, R_j^U and R_j^L as R values, and (17) and (18). These yield the possible final velocities, unless the end of the j'th episode witnesses the jump-stop, in which case $V_{rcr}[1+M_T/M_{rc}]^{-1}$ has to be added, yielding v_j^U and v_j^L.

If $\Delta d_j \leq brDist_j^U$ and $\Delta d_j \leq brDist_j^L$ both fail, then the two stopping distances can be evaluated, as earlier, and the preceding Δd_k values, together with the derived stopping distances, can be accumulated to give the total stopping distances from start of braking. If Δd_j is long enough for only R_j^U to yield braking within the j'th episode, the $j+1$'th episode (and subsequent episodes, if needed) need only consider the R^L calculation(s). The process can then be completed as already described.

7.2. More sophisticated Stochastic Modelling

The previous section considered deterministic but uncertain resistive force. To go beyond that approach, i.e. to entertain greater variability in the resistive force, would entail enhancing the preceding deterministic models with genuine stochastic elements. That, in its turn, would mean extending the hitherto deterministic Hybrid Event-B formalism that we have been using with the capabilities to include such stochastic elements. In the case of mode events this would take us into the realm of probabilistic refinement theory, e.g. [62]. In the case of pliant events, the culmination would be what the mathematicians call stochastic differential equations [64, 40, 53], and what the physicists call Langevin equations [78, 30]. A rigorous development of a fully probabilistic Stochastic Hybrid Event-B incorporating all of these elements remains as work for the future.[15] Moreover, delving more extensively into these questions would derail us too far from the primary objective of this paper, namely the illustration of the manner in

[15]Nevertheless, a partial formalism, featuring stochastic behaviour for pliant events but retaining non/determinism for event scheduling choices and for mode events, has been pioneered in [13].

which impulsive physics can be handled in Hybrid Event-B. For this reason, we do not pursue this line of investigation further.

8. 'Tackling' the Rugby Club Problem

Above, we suggested that if appropriate relationships could be made to hold between the various constants that characterised our simple model (and by implication, for its more complicated successors too), then the Rugby Club Problem might be overcome. In this section, we discuss how the Rugby Club Problem may be addressed when such choices of constants are not available for whatever reason.

In all our models, the principal cause of the loss of coherence between the train's view of the dynamics and the physical reality could be attributed to the fact that the control law for the cruise phase was based exclusively on the train's velocity, whereas the true physics of the situation requires the accounting of momentum.

The obvious suggestion then, would be to change the control laws for the various phases of the dynamics to account for momentum more accurately. In the extremely idealised model of Sections 3 and 4 this would not be hard to do, because in such a simple model, the relationships between velocity and momentum are straightforward, and the cruise phase could easily detect how much momentum it had given away as it brought the train back up to speed. The train could then approach the stopping point more cautiously — knowing firstly that the train is a closed system so that momentum could not be 'lost' in any manner, and secondly that the momentum that appeared to have been mislaid would therefore have to reappear soon.

However, when we consider doing the same thing in the context of the more realistic models contemplated in Section 5, and trialed in Sections 6 and 7, we quickly realise that this is easier said than done. The rugby club steals momentum from the train, but so do all the other sources of non-ideal motion that we mentioned and modelled. Distinguishing between 'natural losses' and 'unnatural losses' becomes highly nontrivial. Therefore, only if natural and unnatural sources of momentum loss can be distinguished clearly enough, could an optional 'more cautious stopping strategy' offer a potential way forward for coping with a mischievous rugby club.

9. Related Work

Hybrid systems have been identified as being of high importance for a long time. The annual *International Conference on Hybrid Systems: Computation and Control*, which is a prominent vehicle for progress in this area, has been running for over twenty years. Some of the earliest work that we can cite includes papers like [61, 6, 7, 45], followed, for example by [60, 37, 41, 74], and slightly later by [59, 24, 46, 51, 28, 36]. The more than decade old survey [27] covers a large number of formulations such as these, and in particular, the tools that support them, such as HyTech [47], d/dt [9], PHaVer [35] and others.

A major consumer of knowledge about hybrid behaviour is, of course, the cyber-physical systems (CPS) field, e.g. [5, 56, 71, 79, 63, 33], as well as the current work presented at the annual *CPS Week* gatherings in recent years. In this context we can point to the extensive survey [38], which covers a wide spectrum of work on cyber-physical systems, as well as the applications that are tackled practically. As we might expect, despite the relative newness of cyber-physical systems, formal approaches are rather overshadowed by more traditional techniques, especially when it comes to practical applications.

Much of the early work cited was characterised by low expressivity in the continuous sphere, motivated, of course, by the desire for decidability. Much work restricted the continuous behaviour to (piecewise) time linear systems. In this context, using a differential equation such as $\mathcal{D}\,x = K$, with K constant, is almost the same as using a linear behaviour such as $x' = x + K\Delta T$ where ΔT is the

duration of the behaviour. Of course, such a severe restriction is very debilitating in the face of many real world problems that fall outside it. To go further means addressing differential equations of a less simple form. However, despite the wealth of knowledge about differential equations and how to solve them [68], the proportion of equations that can be solved analytically is vanishingly small, and the need to tackle practical applications often forces the use of equations for which the only approach is numerical [43, 44].

The fact that it is often not possible to solve a hybrid/cyber-physical system exactly is not the insuperable obstacle it might, at first, seem to be. Often it is sufficient to know that a system will stay in a safe region of the state space indefinitely, without knowing exactly what the system dynamics will be. Terminology differs here. Some authors speak of an 'unsafe region' which is to be avoided. Others speak of a 'safe region' which the system is to be confined to. Still other work speaks of an 'invariant' expression concerning the state space which defines the safe region. Of course, the latter is the terminological domain in which we find Hybrid Event-B and similar systems.

When it is sufficient for the system to stay in a target 'safe region' of the state space, various kinds of 'helper functions' may be employed to gain assurance that the system behaves well.

Variant functions are familiar from the classical discrete programming world [49, 32, 8]. To help control the behaviour of recursions and unbounded iterations, a variant function (of the state) is required to be decreased by each iteration's state change, the idea being that it is easier to ascertain this than to argue about the iterative behaviour directly. When the variant function takes its values in a well founded set, this gives a guarantee of termination, the 'safe region' aimed for being the states in which the iterative behaviour is not enabled.

Liapunov functions are well known from continuous control theory [42, 73, 48]. To help establish stability, the flow defined by the dynamics is required to decrease the Liapunov function (of the state), this being easier to ascertain than to argue about the flow itself, since it can be checked directly from the differential equation defining the flow. The Liapunov function has an easily identified minimum, which coincides with a stable fixed point of the dynamics, the state at which this occurs being the 'safe region' aimed for.

Barrier functions have become a familiar technique for establishing safety in the hybrid systems world [69, 54, 31]. They are required to have one sign (positive say) in the unsafe region, and to have the other sign (negative) in the set of initial states. Provided the barrier function is decreased by the flow defined by the continuous dynamics and is also decreased by each discrete state change, the unsafe region can never be reached. Barrier functions thus combine the basic ideas behind both variant functions and Liapunov functions. Clearly, the rich structure of the hybrid systems paradigm gives rise to many opportunities for fusing ideas from these two precursor worlds.

To profit from the various possibilities for arguing for safety just outlined, the system in question has to be defined precisely enough that rigorous formal reasoning about invariants and barrier functions becomes possible.[16] Among the formalisms designed for such rigour we can mention the following.

One example is the Hybrid CSP system, together with the tools that support it [45, 58, 81]. Another option is the dynamic logic approach of Platzer [66, 67], supported by the KeyMaera verification tool [52] which supports the kind of modelling exemplified in this paper. The original formulation of action systems for discrete systems [11] was extended to the hybrid sphere in [10].

Action systems provided much of the inspiration for the Event-B formalism [2], which builds on the earlier classical B-Method [1]. The mathematical flexibility of the Event-B formalism and the open architecture of its Rodin tool [72, 3] lent themselves to supporting verification of hybrid and cyber-

[16] It has to be said that many systems for CPS in the literature are not defined with sufficient precision to do this, strictly speaking. They avoid trouble by restricting attention to cases that are 'well behaved' in a relatively obvious manner.

physical systems in various ways [76, 75, 26].

These Event-B extensions just cited all adhere to the Event-B paradigm of updating a global description of hybrid behaviour in discrete lumps. More specifically, in each of them there is an *updatable* time variable, and each event execution advances it by appending a lump of behaviour (expressed as a function from a time interval to state values) to the history-so-far. In that specific sense, they differ from the Hybrid Event-B approach of this paper [20, 21], in which time is *read-only*, as stated earlier, and the progress of time is handled by having all relevant state expressions implicitly or explicitly quantified over intervals of time (delegating, if we might be permitted the anthropomorphism, the responsibility for the progress of time to 'God Himself' (in the manner of the normal style of physical discourse)). Broadly speaking, all of these formalisms are capable of supporting the kind of modelling we performed in the more technical parts of this paper — if, in each case, we put aside detailed questions regarding just how well their accompanying mechanised reasoners could cope with the calculations needed, and how much of an overhead their particular syntactic conventions would impose.

10. Summary and Conclusions

In the preceding sections we outlined the essentials of Hybrid Event-B, including enough of the semantic details to facilitate a reasonable grasp of some of the subtle issues needed later, and then we focused on how impulsive physics can be handled in this framework. Then we constructed a Hybrid Event-B model of the rather engaging Rugby Club Problem scenario described in the Introduction. For the purposes of arriving at a reasonably clear exploration of the Rugby Club Problem, our model idealised the situation rather severely. It was thus suffused with point mass and lossless dynamics in the familiar style of classical mechanics. The precision of the model allowed us to derive conditions that distinguished between the non-disruptive and disruptive case of the rugby club dynamics, and we discussed some options for adding more complex invariants to the model. Based on these deliberations, we showed some candidates for the resulting invariants.

We then discussed possibilities for reducing the degree of idealisation in the model, and thus the prospects for making it more realistic, thereby bringing it closer to applicability in practice. In Sections 6 and 7 we described relatively minor enhancements to our simple model along the lines indicated. What is notable about these enhancements is the extreme rapidity with which the calculations involved grew in complexity — this notwithstanding the relative simplicity of the enhancements, and the fact that *all* the consequences of them that we explored could be derived in closed form (albeit that the size of the formulae involved would grow massively). The reader will probably be appreciative of the fact that we suppressed the complicated details.[17]

It is worthwhile, at this point, making an observation about how the stated provability of the additional invariants that were mentioned came about. Most of the analysis of this paper was performed in a fairly *ad hoc* manner. When dealing with a situation described by physical theories, this is, more or less, unavoidable. It follows in turn because physical theories are almost always expressed using a family of equalities. As such, any of the participating variables may (in the given situation) carry input values, with the other variables acquiring their values from the demanded equalities, as outputs. So the derivation process is not structured in a manner that is fixed at the outset, in the way that formal development processes tend to be. However, once the *ad hoc* reasoning has yielded its fruits, we can take a step back, and restructure what has been discovered in a manner that better fits the flow of a formal development process. It is in this manner that the provability that is claimed regarding the additional invariants emerges, and we discussed issues related to this at some length in Section 4.2. Similar remarks would apply in the

[17] Nevertheless, if our goal had been the design of a real urban railway system rather than the illustration of some facets of Hybrid Event-B, we would have had to swallow the ensuing complexity whole.

more complicated case of our enhanced models. In fact, the variations in the derivation process were illustrated quite well in the differing treatments of the stopping problem when varying resistance was assumed to depend, on the one hand on time-defined episodes, and on the other hand on distance-defined episodes.

In the Section 8, we addressed how this modelling exercise could be used to overcome the Rugby Club Problem, in cases where it could not be prevented by choosing appropriate constants. The crux of the matter would be to centre the control system for the train more firmly on the momentum dynamics of the physical system, than on purely kinematic aspects. Confidence in this assertion is supported by the fact that although a rugby club may be able to outwit a train control system whose design is insufficiently suspicious, they cannot cheat the laws of physics. Nevertheless, the suggested approach is only feasible if natural and unnatural sources of momentum depletion can be cleary distinguished, and this is likely to be highly challenging in a realistic situation.

It is instructive to note the very major role played by knowledge of physics in the exercise undertaken in this paper. Although computer scientists often find it convenient to downplay or neglect the influences of non-computing disciplines in the design of systems like ours, which are cyber-physical systems [38, 27],[18] the importance of such influences cannot be denied, and the present exercise shows this eloquently. Cyber-physical systems are truly multidisciplinary and it is unwise to neglect any of the disciplines that contribute to a given system while emphasising just one (e.g. just the computing viewpoint). See [22] for a review of some of the less obvious issues that impact cyber-physical systems, discussed from a mathematical viewpoint.

Acknowledgement The author is delighted to acknowledge discussions with Thiérry Lecomte of ClearSy [29], and with John Baugh, on the subject of this paper.

References

[1] J.R. Abrial, The B-Book: Assigning Programs to Meanings, Cambridge University Press, 1996.

[2] J.R. Abrial, Modeling in Event-B: System and Software Engineering, Cambridge University Press, 2010.

[3] J.R. Abrial, M. Butler, S. Hallerstede, T. Hoang, F. Mehta, L. Voisin, Rodin: An Open Toolset for Modelling and Reasoning in Event-B, Int. J. Soft. Tools Tech. Trans. 12 (2010) 447–466.

[4] R. Banach et al., Core Hybrid Event-B III: Fundamentals of a Reasoning Architecture (in preparation).

[5] R. Alur, Principles of Cyberphysical Systems, MIT Press, 2015.

[6] R. Alur, C. Courcoubetis, T. Henzinger, P.H. Ho, Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems, in: Proc. Workshop on Theory of Hybrid Systems, volume 736 of *LNCS*, Springer, 1993, pp. 209–229.

[7] R. Alur, D. Dill, A Theory of Timed Automata, Theor. Comp. Sci. 126 (1994) 183–235.

[8] K. Apt, Ten Years of Hoare's Logic: A Survey Part I, A.C.M. Trans. Prog. Lang. Sys. 3 (1981) 431–483.

[18] See, for example, the balance of content in references such as [56, 5].

[9] E. Asarin, T. Dang, O. Maler, The d/dt Tool for Verification of Hybrid Systems, in: Proc. CAV-02, volume 2404, Springer, LNCS, 2002, pp. 365–370.

[10] R.J. Back, L. Pete, I. Porres, Generalising Action Systems to Hybrid Systems, in: Proc. FTRTFT-00, volume 1926, Springer, LNCS, 2000, pp. 202–213.

[11] R.J. Back, J. von Wright, Refinement Calculus: A Systematic Introduction, Springer, 1998.

[12] R. Banach, Pliant Modalities in Hybrid Event-B, in: Liu, Woodcock, Zhu (Eds.), Proc. Jifeng He Festschrift 2013, volume 8051 of *LNCS*, Springer, 2013, pp. 37–53.

[13] R. Banach, Contemplating the Addition of Stochastic Behaviour into Hybrid Event-B, in: Wang, Leucker (Eds.), Proc. IEEE TASE-14, pp. 42–49.

[14] R. Banach, Formal Refinement and Partitioning of a Fuel Pump System for Small Aircraft in Hybrid Event-B, in: Bonsangue, Deng (Eds.), Proc. IEEE TASE-16, pp. 65–72.

[15] R. Banach, Hemodialysis Machine in Hybrid Event-B, in: Butler, Schewe, Mashkoor, Biro (Eds.), Proc. ABZ-16, volume 9675, Springer, LNCS, 2016, pp. 376–393.

[16] R. Banach, The Landing Gear System in Multi-Machine Hybrid Event-B, Int. J. Soft. Tools for Tech. Trans. 19 (2017) 205–228.

[17] R. Banach, Issues in Automated Urban Train Control: 'Tackling' the Rugby Club Problem, in: Butler, Raschke (Ed.), Proc. ABZ-18, volume 10817, Springer, LNCS, 2018, pp. 171–186.

[18] R. Banach, M. Butler, A Hybrid Event-B Study of Lane Centering, in: Aiguier, Boulanger, Krob, Marchal (Eds.), Proc. CSDM-13, Springer, 2013, pp. 97–111.

[19] R. Banach, M. Butler, Cruise Control in Hybrid Event-B, in: Liu, Woodcock, Zhu (Eds.), Proc. ICTAC-13, volume 8049 of *LNCS*, Springer, 2013, pp. 76–93.

[20] R. Banach, M. Butler, S. Qin, N. Verma, H. Zhu, Core Hybrid Event-B I: Single Hybrid Event-B Machines, Sci. Comp. Prog. 105 (2015) 92–123.

[21] R. Banach, M. Butler, S. Qin, H. Zhu, Core Hybrid Event-B II: Multiple Cooperating Hybrid Event-B Machines, Sci. Comp. Prog. 139 (2017) 1–35.

[22] R. Banach, W. Su, Cyberphysical Systems: A Behind-the-Scenes Foundational View, in: Mashkoor, Thalheim, Wang (Eds.), Proc. Klaus-Dieter Schewe Festschrift 2018, volume 34, College Publications, Tribute Series, 2018, pp. 177–201.

[23] R. Banach, P. Van Schaik, E. Verhulst, Simulation and Formal Modelling of Yaw Control in a Drive-by-Wire Application, in: Proc. FedCSIS IWCPS-15, pp. 731–742.

[24] Bender, K. and Broy, M. and Péter, I. and Pretschner, A. and Stauner, T., Model Based Development of Hybrid Systems: Specification, Simulation, Test Case Generation, in: Modelling, Analysis, and Design of Hybrid Systems, volume 279, Springer, LNCS, 2002, pp. 37–51.

[25] A. Bloch, P. Krishnaprasad, R. Murray, J. Baillieul, P. Crouch, J. Marsden, D. Zenkov, Nonholonomic Mechanics and Control, Springer, 2015.

[26] M. Butler, J.R. Abrial, R. Banach, Modelling and Refining Hybrid Systems in Event-B and Rodin, in: From Action System to Distributed Systems: The Refinement Approach, CRC Press, 2016, pp. 29–42.

[27] L. Carloni, R. Passerone, A. Pinto, A. Sangiovanni-Vincentelli, Languages and Tools for Hybrid Systems Design, Foundations and Trends in Electronic Design Automation 1 (2006) 1–193.

[28] A. Cimatti, M. Roveri, Requirements Validation for Hybrid Systems, in: Proc. CAV-09, volume 5643, Springer, LNCS, 2009, pp. 188–203.

[29] ClearSy, http://www.clearsy.com/.

[30] W. Coffey, Y. Kalmykov, The Langevin Equation, World Scientific, 2017. 4th ed.

[31] L. Dai, T. Gan, B. Xia, N. Zhan, Barrier Certificates Revisited, J. Symb. Comp. 80 (2017) 62–86.

[32] E. Dijkstra, A Discipline of Programming, Prentice-Hall, 1976.

[33] A. Egyed, A Roadmap for Engineering Safe and Secure Cyber-Physical Systems, in: Proc. MEDI-18 Workshops, volume 929, Springer, CCIS, 2018, pp. 113–114. Invited talk.

[34] A. Fasano, S. Marmi, Analytical Mechanics, Oxford University Press, 2013.

[35] G. Frehse, PHaVer: Algorithmic Verification for Hybrid Systems past HyTech, Int. J. Tools Tech. Trans. 10 (2008) 263–279.

[36] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, O. Maler, SpaceEx: Scalable Verification of Hybrid Systems, in: Proc. CAV-11, volume 6806, Springer, LNCS, 2011, pp. 379–395.

[37] V. Friesen, A. Nordwig, M. Weber, Object-Oriented Specification of Hybrid Systems using UML, h and ZimOO, in: Proc. ZUM-98, volume 1493, Springer, LNCS, 1998, pp. 328–346.

[38] E. Geisberger, M. Broy (eds.), Living in a Networked World. Integrated Research Agenda Cyber-Physical Systems (agendaCPS), 2015. http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Publikationen/Projektberichte/acaetch_STUDIE_agendaCPS_eng_WEB.pdf.

[39] H. Goldstein, C. Poole, J. Safko, Classical Mechanics, Addison Wesley, 2001.

[40] G. Grimmett, D. Stirzaker, Probability and Random Processes, Oxford University Press, 2001. 3rd ed.

[41] Grosu, R. and Stauner, T. and Broy, M., A Modular Visual Model for Hybrid Systems, in: Proc. FTRTFT-98, volume 1486, Springer, LNCS, 1998, pp. 75–91.

[42] W. Haddad, V. Chellaboina, Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach, Princeton University Press, 2008.

[43] E. Hairer, S. Norsett, G. Wanner, Solving Ordinary Differential Equations I Nonstiff Problems, Springer, 1993.

[44] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II Stiff and Differential-Algebraic Problems, Springer, 1996.

[45] J. He, From CSP to Hybrid Systems, in: Roscoe (Ed.), A Classical Mind, Essays in Honour of C.A.R. Hoare, Prentice-Hall, 1994, pp. 171–189.

[46] T. Henzinger, The Theory of Hybrid Automata, in: Proc. IEEE LICS-96, IEEE, 1996, pp. 278–292. Also http://mtc.epfl.ch/~tah/Publications/the_theory_of_hybrid_automata.pdf.

[47] T. Henzinger, P.H. Ho, H. Wong-Toi, HyTech: A Model Checker for Hybrid Systems, Int. J. Tools Tech. Trans. 1 (1997) 110–122.

[48] D. Hinrichsen, A. Pritchard, Mathematical Systems Theory I, Springer, 2005.

[49] C. Hoare, An Axiomatic Basis for Computer Programming, Comm. A.C.M. 12 (1969) 576–580.

[50] J. Horvarth, Topological Vector Spaces and Distributions, Dover, 2012.

[51] Y. Kesten, Z. Manna, A. Pnueli, Verification of Clocked and Hybrid Systems, Acta Informatica 36 (2000) 837–912.

[52] KeYmaera. http://symbolaris.com.

[53] P. Kloeden, E. Platen, Numerical Solution of Stochastic Differential Equations, Springer, 1992. 4th ed.

[54] H. Kong, F. He, X. Song, W. Hung, M. Gu, Exponential-Condition-Based Barrier Certificate Generation for Safety Verification of Hybrid Systems, in: Proc. CAV-13, volume 8044, Springer, LNCS, 2002, pp. 242–257.

[55] T. Lecomte, Atelier B has Turned 20, in: Proc. ABZ-16, volume 9675, Springer, LNCS, 2016, p. XVI.

[56] E. Lee, S. Shesha, Introduction to Embedded Systems: A Cyberphysical Systems Approach, LeeShesha.org, 2nd. edition, 2015.

[57] M. Leuschel, M. Butler, ProB: An Automated Analysis Toolset for the B Method, Int. J. Soft. Tools Tech. Trans. 10 (2008) 447–466.

[58] J. Liu, J. Lv, Z. Quan, H. Zhao, C. Zhou, L. Zou, A Calculus for Hybrid CSP, in: Ueda (Ed.), Proc. APLAS-10, volume 6461, Springer, LNCS, 2010, pp. 1–15.

[59] N. Lynch, R. Segala, F. Vaandrager, Hybrid I/O Automata, Information and Computation 185 (2003) 105–157. MIT Technical Report MIT-LCS-TR-827d.

[60] N. Lynch, R. Segala, F. Vaandrager, H. Weinberg, Hybrid I/O Automata, Springer, 1996.

[61] Z. Manna, A. Pnueli, Verifying Hybrid Systems, in: Hybrid Systems, volume 736, Springer, LNCS, 1993, pp. 4–35.

[62] A. McIver, C. Morgan, Abstraction, Refinement and Proof for Probabilistic Systems, Springer, 2004.

[63] National Science and Technology Council, Trustworthy Cyberspace: Strategic plan for the Federal Cybersecurity Research and Development Program, 2011. http://www.whitehouse.gov/sites/default/files/microsites/ostp/fed_cybersecurity_rd_strategic_plan_2011.pdf.

[64] B. Oksendal, Stochastic Differential Equations: An Introduction with Applications, Springer, 2013. 6th ed.

[65] J. Papastavridis, Analytical Mechanics: A Comprehensive Treatise on the Dynamics of Constrained Systems, World Scientific, 2nd. edition, 2014.

[66] A. Platzer, Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics, Springer, 2010.

[67] A. Platzer, Logical Foundations of Hybrid Systems, Springer, 2018.

[68] A. Polyanin, V. Zaitsev, Handbook of Ordinary Differential Equations: Exact Solutions, Methods, and Problems, C.R.C. Press, 2018.

[69] S. Prajna, A. Jadbabaie, Safety Verification of Hybrid Systems Using Barrier Certificates, in: Proc. HSCC-04, volume 2289, Springer, LNCS, 2004, pp. 477–492.

[70] ProB Tool. https://www3.hhu.de/stups/prob/.

[71] C. Ptolemaeus (ed.), System Design, Modeling, and Simulation Using Ptolemy II, Ptolemy.org, 2014.

[72] RODIN Tool. http://www.event-b.org/ http://sourceforge.net/projects/rodin-b-sharp/.

[73] E. Sontag, Mathematical Control Theory, Springer, 1998.

[74] T. Stauner, B. Rumpe, P. Scholz, Hybrid System Model. Technische Universitat Munchen, TUM-I9903.

[75] W. Su, J.R. Abrial, Aircraft Landing System: Approaches with Event-B to the Modelling of an Industrial System, Int. J. Soft. Tools Tech. Trans. 19 (2017) 141–166.

[76] W. Su, J.R. Abrial, H. Zhu, Formalising Hybrid Systems with Event-B and the Rodin Platform, SCi. Comp. Prog. 94 (2014) 164–202.

[77] F. Treves, Topological Vector Spaces, Distributions and Kernels, Dover, 2007.

[78] N. Van Kampen, Stochastic Processes in Physics and Chemistry, Elsevier, 2007. 3rd ed.

[79] W. Wolf, Cyber-Physical Systems, IEEE Computer 43 (2009) 88–89.

[80] A. Zemanian, Distribution Theory and Transform Analysis: An Introduction to Generalized Functions, with Applications, Dover, 2003.

[81] N. Zhan, S. Wang, H. Zhao, Hybrid CSP, in: Formal Verification of Simulink/Stateflow Diagrams: A Deductive Approach, Springer, 2017, pp. 71–90.