Towards Dynamic Partitioning of States of a Reactive System: Train Controller Case Study

Victor Winter Deepak Kapur

September 4, 2005

1 Overview

Based on our investigations of a case study of controllers for train systems [1, 2, 3, 4], we present a model of reactive systems which emphasizes *dyanmic* partitioning of system states into *normal* and *abnormal*, to respond to nondeterministic events due to environment factors. In this sense, reactive systems under consideration are non-strict that their behavior is not completely governed by past events; instead, unexpected future events must also be considered in the design of controllers for such systems.

A reactive system can be usually modelled using a set of monitored and controlled variables, and their values at any given instance determines the *state* of the system. A *transition function* on states is defined capturing how the system state changes under the effect of changing values of one or more controlled variables. In this context, the *control problem* entails the discovery of a sequence of allowable assignments to the controlled variables such that the resulting system behavior satisfies a desired set of constraints, often couched in terms of a safety policy and additional secondary requirements focussing on throughput. However, environmental events can take place forcing unexpected change in the value of one or more controlled variables.

Our key observation is that system states should not be treated alike (e.g., a system state arising due to a derailment of a train(s)) and instead, they should be partitioned into normal and abnormal states using some criterion related to the *expected* behavior of a given reactive system. States arising due to the expected behavior are normal whereas other states are abnormal. Further, this partitioning is dynamic as one part of a reactive system may be exhibiting expected normal behavior whereas another part of the reactive system is already behaving unexpectedly. A case in point in a train system is when a train abruptly stops because its track is observed to be blocked, or in a worst case, a train derails. Depending upon the abnormal behavior of the train, the trains following it may continue to behave normally or may have to resort to drastic "abnormal" actions (such as trailing trains continuing cautiously hoping that the abnormal situations will be fixed until a certain distance after which it must stop). In this sense, it is reasonable to allow trailing trains to undertake action (in terms of change in their accelerations) that is more extreme than normal, to avoid collision. Actions in normal states are thus in general a subet of actions in abnormal states since abnormal states arising due to unexpected environmenal events may require drastic actions, such as slamming the brakes.

Adopting a **strict reactive classification scheme** may be too rigid for systems in which nondeterministic events leading to abnormal states are extremely rare and/or the normal capabilities of the system are significantly restricted. It may be worthwhile instead to consider a **non-strict** classification scheme that enables a greater number of behaviors and thus gives the control function more flexibility in how it can go about satisfying its requirements. In turn, this flexibility may make it possible for the control function to more satisfactorily address competing sets of requirements.

2 Basic Model and Application

As stated above, our insights are based on designing controllers for a complex train system populated by several trains, tracks, switches, signals and stations, similar to the BART system. We are interested in controlling the behavior of an individual train by developing a software function capable of controlling its behavior in a manner that satisfies a given *safety policy*. A number of secondary requirements are in direct competition with the requirements in the safety policy including: throughput optimization, passenger comfort, and energy conservation.

The proposed model is simple in the sense that control over a train is exercised by changing its acceleration: negative accelerations cause the train to slow down and eventually stop (but not move backwards); positive accelerations cause the train to increase its speed. Acceleration values are drawn from a single set $\mathcal{A}$. Under normal conditions, acceleration values may only be drawn from a set $\mathcal{A}_N$ where $\mathcal{A}_N \subset \mathcal{A}$.

2.1 The Basic Model

The basic model of a train system is given as a family of models for each train described using the following components:

$$\{state, profile, next\_state, list\_mu, \ll\}.$$

Below, we briefly review the role of each of these components informally. More details can be found in [1, 2, 3, 4].

- A *state* of a given train at any given time is modelled by a triple of the form $(position, speed, acceleration)$.
- A *profile* is modelled as a sequence of train states, thus describing the behavior of a train over a given time interval.
- The *next_state* function is an instance of the transition function defined over individual trains. For a given state st_i at time instance i of a given train and its acceleration value $a \in \mathcal{A}$, *next_state* returns a new train state st_{i+1} at time instance $i+1$ as its output.
- The function *list_mu* is used to generate profiles through the repeated application of the *next_state* function to study different scenarios. A controller can use this ability to speculate on the possible future behaviors of a train, including those which optimally capture the assumptions implied by the constraints of a given safety policy as well as those that represent worst-case scenarios.
- The binary relation $\ll$ over profiles capture profiles that can safely coexist (i.e., a collision will not occur). For example, let (t_{ot}, t_{lt}) denote a pair of adjacent trains, where t_{ot} is an object train with t_{lt} as its corresponding lead train. If $pf_{ot} \ll pf_{lt}$ holds, where pf_{ot} and pf_{lt}, respectively denote the profiles of an object train and its lead train, then the behaviors of (t_{ot}, t_{lt}) are collision-free with respect to one another.

The main goal of a controller for the object train t_{ot} is to ensure the following:

1. For every possible future profile pf_{lt} of the lead train, a corresponding future profile pf_{ot} can be generated such that $pf_{ot} \ll pf_{lt}$.

3 Nondeterminism and Safety Assumptions

When modelling a train system, nondeterminism results from numerous environmental sources. For example, a train may derail unexpectedly or the train operator may engage the emergency brake of a train due to an unexpected situation, like an automobile stalled on the track. *Model drift* also represents a tacit source of nondeterminism since the model of train behavior is an approximation and without corrective feedback, the actual behavior of the train can, over time, diverge from the behavior predicted by its model.

Nondeterminism in the above model raises the question as to what assumptions can be reasonably made regarding the future behavior of a train. Safety requirements are typically met by conservatively bounding the behavior of train models. Under a set of conservative assumptions, the controller of an object train t_{ot} must be prepared to respond to a set of worst-case behaviors of its lead train t_{lt} in a manner consistent with the stated safety policy. For example, when the system is operating under normal conditions, the worst-case behavior of t_{lt} would be to come to a halt as quickly as permitted by the constraints placed on its normal operation. On the other hand, when the system is displaying nondeterministic behavior, the worst-case behavior of t_{lt} could be that it derails, i.e., itinstantaneously comes to a dead halt through a derailment.

Let d_N denote the shortest distance needed by a train to stop using acceleration values taken only from $\mathcal{A}_N \subseteq \mathcal{A}$ and d_E be the shortest distance needed by a train to stop using acceleration values taken from $\mathcal{A}$. The distance $d_E \leq d_N$. When d_E is significantly smaller than d_N, there is a certain amount of *play* (flexibility) in the system. I.e., train behaviors can be based on less conservative assumptions, secure in the

knowledge that, in the worst-case, there will still be plenty of opportunity to avoid a collision through the use of the emergency brake. Another way of looking at this phenomenon is that the gap between d_E and d_N may, under certain assumptions, reduces the period of the corrective feedback cycle. We introduce the notion of a *profile buffer bf* that describes the predicted future behavior of a train over a given interval under deterministic circumstances. If all goes as expected, the behavior of a train t_{lt} will correspond to its profile buffer bf_{lt}. Figure 1 graphically depicts how profile buffers can be used to extend the onset of pessimistic behavior resulting from conservative bounding. When the actual behavior of the lead train t_{lt} diverges from the predicted behavior of its profile buffer, it becomes possible for the object train t_{ot} to react using the full range of accelerations in $\mathcal{A}$.

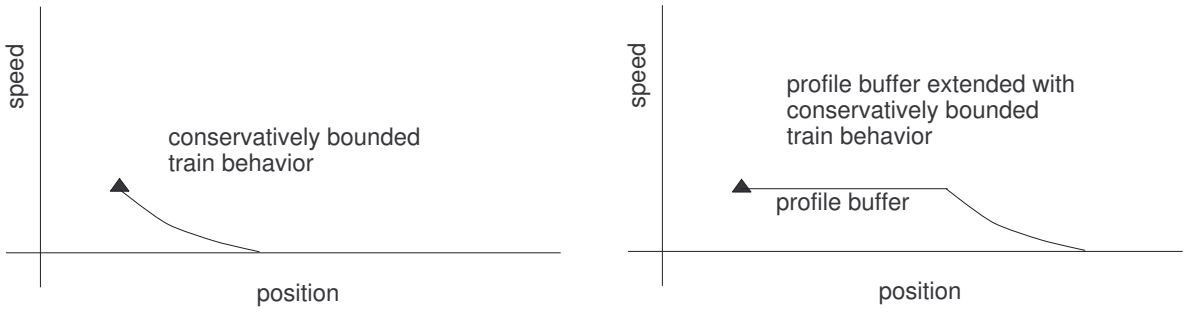

Figure 1: Using a profile buffer to delay the onset of conservative bounding of behavior

References

[1] D. Kapur and V. L. Winter. On the construction of a domain language for a class of reactive systems. In V. Winter and S. Bhattacharya, editors, *High Integrity Software*, pages 169–196. Kluwer Academic Publishing, 2001.

[2] D. Kapur, V. L. Winter, and R. S. Berg. Designing a controller for a multi-train multi-track system. In C. Zaroliagis, editor, *Electronic Notes in Theoretical Computer Science (ENTCS)*, volume 50. Elsevier Science Publishers, 2001.

[3] V. L. Winter, D. Kapur, and R. S. Berg. A refinement-based approach for developing software controllers for train systems. In V. Winter and S. Bhattacharya, editors, *High Integrity Software*, pages 197–240. Kluwer Academic Publishing, 2001.

[4] V. L. Winter, D. Kapur, and G. Fuehrer. Formal specification and refinement of a safe train control function. In F. Kordon and M. Lemoine, editors, *Formal methods for embedded distributed systems : how to master the complexity*. Kluwer Academic Publishing, 2004.