Proceedings of the
Ninth International Workshop on
Automated Verification of Critical Systems
(AVOCS 2009)

Specifying Railway Interlocking Systems

Karim Kanso Anton Setzer

3 pages

Specifying Railway Interlocking Systems[*]

Karim Kanso **Anton Setzer**

Dept. Computer Science, Swansea University, Swansea, SA2 8PP, UK
email: {cskarim, A.G.Setzer}@swansea.ac.uk

Abstract: One of the Grand Challenges in Computer Science is to verify railway interlocking systems [1]. We give a generic datatype of control tables and ladder logic (2,3), and extract from these verification conditions (4). A proof of the correctness of these conditions is performed using induction and a datatype of reachable states (5). Finally, some concluding remarks are presented (6). This specification has been implemented in Agda2.

Keywords: verification, specification, dependent types, Agda2, railway, control system, interlocking, ladder logic

Previously, [3] we developed a verification system that applied SAT solver technology to verify arbitrary first order formulæ w.r.t. safety. This work builds upon previous work by formalising what a control table is and what it means for an interlocking system to ratify a verification condition.

1 Physical Layout

Initially, when designing a railway, it is required to fix a physical layout of the involved hardware, i.e. track segments, signals, platforms, emergency systems, etc. These pieces of hardware have attributes, i.e. track segments can be occupied or unoccupied and signals can be green or red. Many techniques have been used to formally model the physical layout of a railway [4]. Layouts must contain all identifiers of the components along with their relationships; they are viewed as a signature for the interlocking. Let *Layout* be the type of physical layouts, and $p : Layout$.

2 Control Tables

Control tables are used to define the behaviour of the hardware in traditional railway signalling systems; they are a set of rules/constraints that must be observed and are an abstract specification for a portion of the railway. Responsibility of the safety of the railway is delegated to control tables, which express for instance that a green signal is only shown when a given constraint is met.

Control tables are sentences built over a physical layout, in our system we define a control

[*] This research is funded by Westinghouse Rail Systems, Chippenham, UK

table to be a list of relations, which formalise the constraints. Relevant signatures are:

data $ControlTableEntry_p$ where
$route : RouteID \to Signal \to Signal \to List(Track) \to ControlTableEntry_p$
$<$ other relations $>$

$$ControlTable_p = List\ ControlTableEntry_p$$

An entry $route(r, s_0, s_1, [t_1, \ldots, t_n])$ expresses that route r starts at signal s_0, ends at signal s_1, and uses track segments $t_1, \ldots, t_n$.

3 Ladder Logic

Interlocking systems are realised using a multitude of techniques; systems programmed using ladder logic are the focus of this research. Ladder logic is a discrete time, linear system of Boolean equations, relating Boolean valued inputs and internal state to Boolean valued outputs. Ladder logic programs are represented using a transition function $next : Internal \times Input \to Internal$ between states, an output function $output : Internal \times Input \to Output$ and an initial internal state. In the following, let $Ladder$ be the type of ladder logic programs, and $l : Ladder$; $Internal$, $Input$ and $Output$ are indexed by l. Notably, l is a model of a control table c *iff* l never violates the constraints in c.

4 Verification Conditions

To determine whether a ladder logic program correctly refines a control table, verification is required. Firstly, the datatype of correctness is defined as a relation between the input and output of the system; $Correctness_l \subseteq Input_l \times Output_l$. Secondly, the verification condition type is defined as a function from a control table entry to $Correctness_l$.

$$VerifCond_p^l = ControlTableEntry_p \to Correctness_l$$

An intersection between multiple correctness relations is used to construct combined correctness relations. This technique can be used to verify all the control table.

Verification conditions are sentences built over a physical layout; they can be generated from the relations in the control table by instantiating a template sentence. E.g. for a signal with only one route, such as exist in some installations the *route* relation can be mapped to a correctness relation which states “if a track segment in the route is occupied, then the first signal shows a red aspect”.

$$\begin{aligned}
&f : VerifCond_p^l \\
&f(route(rt, s_1, s_2, ts))(in, out) \Leftrightarrow \\
&\quad fold(\vee, false, map((\lambda x \bullet in(x.occupied)), ts)) \to out(s_1.red)
\end{aligned}$$

Correctness conditions can be independent from the control table entries; this is particularly useful when verifying general safety properties. E.g. a signal does not display both red and green aspects at the same time[1].

[1] In practice the signalling policy in use could allow for a transitional period where both aspects are shown.

5 Correctness

Verification uses the principle of induction, working from an initial internal state at time 0 up to time n. States might be unreachable causing false negatives during the verification. This problem is avoided by using a datatype of reachable states.

Reachable states are defined using induction-recursion [2]. The initial state is reachable by definition; from any reachable state at time t, after processing the inputs, state $t+1$ is also reachable. We obtain a type $ReachableState_l$ and a function $toInternal_l : ReachableState_l \to Internal_l$. A proof of correctness is then a proof that the relation is not empty, i.e.

$$correct_l^q : (r : ReachableState_l) \to (i : Input_l) \to q(i, output_l(toInternal_l(r), i))$$

where $q : Correctness_l$.

6 Concluding Remarks & Future Work

The above scheme has been implemented using the theorem prover and programming language Agda2[2], which is based on constructive type theory (intuitionistic logic). We have verified simple toy examples; to verify complex problems we intend to explore the possibility of linking Agda2 to a SAT solver. Also, a CASL implementation is planned.

We conjecture that an institution can be defined where the signatures are physical layouts (plus a chosen logic signature), sentences are control tables (moreover a sentence is a formula built using the chosen logic), models are ladder logic programs and the satisfaction relation is derived from the correctness proof.

Bibliography

[1] Bjørner, D., *TRain: The Railway Domain*, in: *Building the Information Society*, IFIP International Federation for Information Processing **156/2004** (2004), pp. 607–611.
URL http://www.springerlink.com/content/527p7237102w5741/

[2] Dybjer, P. and Setzer, A., *Indexed Induction-Recursion*, in: *Proof Theory in Computer Science*, LNCS Lecture Notes in Computer Science **2183** (2001), pp. 93–113.
URL http://www.springerlink.com/content/rc3t3m7gkfnmpq3d/

[3] Kanso, K. and Moller, F. and Setzer, A., *Automated Verification of Signalling Principles in Railway Interlocking Systems*, to appear in: ENTCS, (2009)

[4] Pnika, M., *Formal Approach to Railway Applications*, in: *Formal Methods and Hybrid Real-Time Systems*, LNCS Lecture Notes in Computer Science **4700** (2007), pp. 504–520.
URL http://www.springerlink.com/content/c82275854v4k29q0/

[2] http://wiki.portal.chalmers.se/agda/