Verifying Train Control Software – Using SAT-based Model Checking.

Phillip James

Department of Computer Science
Swansea University, United Kingdom

April 21, 2010

In co-operation with Invensys.
Overview

• Verification Within The Railway Domain.
• Reachable State Algorithms.
• Example Application.
• Ideas On Tackling The State Space Explosion.
Verification Within The Railway Domain
Motivation

Metrolink passenger train collides with freight train.

25 people killed, over 100 people injured!
A major system responsible for ensuring railway safety is the railway interlocking.

- Interlockings control aspects such as signals and points.
- Interlockings are written using a logic language similar to propositional logic.
Railway Verification in Propositional Logic – Kanso 2008

Railway Topology

Informal Safety Condition

Interlocking Ladder Logic

\[ \varphi(\mu) \]

Safety Condition

\[ \mu, \mu' : \text{Represent system states.} \]

(i.e. - Valuations of propositional variables)

Successful Automated Verification:

1. \( \neg(I(\mu) \Rightarrow \varphi(\mu)) \)
2. \( \neg(\varphi(\mu) \land T(\mu, \mu') \Rightarrow \varphi(\mu')) \)

If 1 & 2 are not satisfiable then output “safe”
Problems with Kanso '08:

- Often there are violations of $\neg(\varphi(\mu) \land T(\mu, \mu') \rightarrow \varphi(\mu'))$ that are unreachable.

- Approach leads to many unreachable counter examples – “Not Safe” is returned when in fact program is correct.
Our Aims

Our aims:

- Devise a verification method which ignores unreachable states.
- If a counterexample is found, produce an error trace to the counterexample.
- Implement these techniques into a useable verification tool which works on real world interlockings.
Our Approach
Addressing Reachability

Forwards Reachability in K Steps – Sheeran et al

\[
i \leftarrow 0 \\
B_0 \leftarrow \{\mu \mid I(\mu)\} \\
do \\
\quad \quad \quad B_{i+1} \leftarrow \{\mu' \mid T(\mu, \mu')\} \\
\quad \quad \quad \text{for } \mu \in B_{i+1}, \text{ if } \neg(\phi(\mu)) \in SAT \text{ return trace} \\
\quad \quad \quad i \leftarrow i + 1 \\
while i \leq K \\
\text{return "K-Safe"}
\]

Eliminates unreachable states problem – Only states reachable from the initial state of the system are verified.
Pelican Crossing Example
A Pelican Crossing

Verification Within The Railway Domain
Our Approach
Pelican Crossing Example
State Space Explosion

Phillip James
Verifying Train Control Software
Specifying in Hets

Pelican Crossing Ladder Logic (Transition Formula)

```
spec Transition [State0][State1] =
  . crossing1 <=> req0 \ not crossing0
  . req1 <=> pressed0 \ not req0
  . tlag1 <=> not crossing1 \ (not pressed0 \ req1)
  . tblg1 <=> not crossing1 \ (not pressed0 \ req1)
  . tlar1 <=> crossing1
  . tlbr1 <=> crossing1
  . plag1 <=> crossing1
  . plbg1 <=> crossing1
  . plar1 <=> not crossing1
  . plbr1 <=> not crossing1
  . audio1 <=> crossing1
end
```
Kanso Approach - Verification Wrongly Fails

Verification:
Kanso 2008

Not Safe – Due to unreachable counter examples.
Our Approach Verification Successful

- Pelican Crossing Topology
- Informal Safety Condition
- Ladder Logic

Verification: Forward K-Step Algorithm

3-Safe
Meta reasoning: Safe!
Hets & Minisat

Always green or red...
(tlar V tlag) \(\land\) (tlbr V tlbg)
Safety Condition

Phillip James Verifying Train Control Software
Tool Example.
State Space Explosion
Insights Gained

- Only a fraction of complete state space is reachable.

- This should help greatly on larger examples (\(2^{12}\) states in example, \(2^{300}\) for interlockings).

- Possible to make whole process automatic by adding state inclusion tests.
Methods Of State Space Reduction

- Remove variables that depend on similar values. E.g. if $X_3, X_4 := \neg X_1 \land X_2$.

- Exclude invariants (Physical and Encoding). E.g. 3 valued data encoded in two bits.

- Slicing transition formula, relative to safety condition. E.g. only include parts of ladder logic that safety condition depends on.
Forwards reachability approach works well on simple examples:
- Eliminating problem of unreachable violating states,
- Produces error traces.

We plan to...
- Implement backwards reachability algorithm.
- Explore performance on real world problems (train control).
- Study slicing methods to improve any performance issues.