Languages, Models, Analysis, and Comprehension

Victor Winter
Outline

My Area of Expertise

Reactive Systems

The BART Case Study
  History of Case Study
  System Overview

Our Research
Transformation provides a useful conceptual framework for studying languages.
What is Transformation?

- The manipulation of artifacts.
  - Documents
  - Specifications
  - Programs
  - Models

- Necessary precondition: The structure of the artifacts in the domain of discourse must be described in a formal fashion.
  - Specification Languages
  - Programming Languages
  - Domain-specific Languages
HATS - A Transformation System

- A tool developed in-house to support language-centric research.

- Features
  - Powerful Syntax Capabilities
    - Support for extended-BNF notation including precedence and associativity.
    - GLR Parser
    - Sophisticated pretty-printing capabilities
  - Powerful Analysis and Manipulation Capabilities
    - A special purpose language, called TL, for developing meta-programs.
Java Library Adaptation

Examples of Transformation in Practice

A Java Classloader


A Transformational Perspective into the Core of an Abstract Class Loader for the SSP.

Examples of Transformation in Practice

Generation of a Train Control Function

Victor L. Winter, F. Kordon, and M. Lemoine.


Components of a Typical Reactive System

- **Sensors** – provides information about the state of the system
- **Actuators** – used to change the state of the system
- **Control Function** – used to calculate how the state should be changed
The Classical Sense/React Loop

1. **Get** information from sensors

2. **Compute** the system’s response to the sensor information

3. **Send** appropriate commands to actuators

4. **Repeat**
The Need for Continuous Adjustment

- **Sensor information**
  - does not provide a complete description of the state
  - has limited resolution

- **Control is computed based on**
  - a model of the system (not the system itself)
  - a perception of state based on initial state, sensor input, and history

- **Actuator behavior**
  - has limited capabilities
  - has limited resolution
The following properties must hold in order for a reactive system to function properly:

- States requiring incompatible adjustments must be distinguishable.
- The sense/react loop must be fast enough to keep up with the rate of change of the system.
- Actuators must be sufficiently refined so that desired state transitions can be achieved.
Inception

- In the 90’s, a project was underway to increase the throughput of the Bay Area Rapid Transit (BART) system by spacing trains closer together (approximately 30 seconds closer).

- In the proposed system, trains would be controlled by computers – a significant departure from the previous design.

- Sandia National Laboratories was engaged to perform a safety analysis of the proposed system.

- This gave Sandia access to the details of the project from which the BART case study was created.
History

- **HIS 1999** – Theme of the High-Integrity Software (HIS) conference organized by Sandia National Laboratories.

- **ICSE 2000** – Axel van Lamsweerde devotes a major portion of his keynote address to the Bart case study where he refers to it as the “current benchmark for FMs”.

- **Dagstuhl 2001** – Theme of a week-long seminar at Dagstuhl titled “Can Formal Methods Cope with Software-Intensive Systems?”

- **Monterey Workshop 2005** – Challenge Problem.
My Area of Expertise

Reactive Systems

The BART Case Study

History of Case Study

System Overview

Our Research
Technical Details

- System
  - 26 control stations
  - 72 miles of double track
  - 80 trains

- Capacity
  - BART serves around 250,000 passengers per day.
  - During normal operation 50 trains consisting of 10 cars are typically in service.
Trains

- Cars are driven by electric motors powered by a 1000 VDC third rail.
- Cars use both regenerative and friction brakes.

Track

- The track system is acyclic.
- A double track configuration enables bidirectional travel between stations.
- A track is partitioned into track segments.
- Speed limits are associated with track segments.

Interlocking System

- Consists of switches and gates.
- Switches control train routes.
- Gates are used to control flow of traffic and right-of-way.
Automated Control
- The speed and acceleration of trains is controlled automatically by the Advanced Automatic Train Control (AATC) system.
- A station computer is used to calculate and broadcast control commands to trains in a given region.
- A station computer is expected to control at most 20 trains.
- Train commands are broadcast over a radio communications network.

Manual Control
- Signaling the system when platforms are clear.
- Trouble-shooting problems.
- Operating trains at low speeds when there is a problem.
Communication

- Control commands are issued every 0.5 seconds.
- A computer on the lead car of each train carries out the control commands received from station computers.
- Commands are time stamped and expire after 2 seconds at which time the train will go into maximum braking mode.
- A train will continue to carry out a command until:
  - a new command is received, or
  - the current command expires.
## Sample Track Data

<table>
<thead>
<tr>
<th>Segment</th>
<th>Begin (feet)</th>
<th>End (feet)</th>
<th>Length (feet)</th>
<th>Comment</th>
<th>Civil Speed (mph)</th>
<th>Grade</th>
<th>Exposure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Civil Speed</td>
<td>5940</td>
<td>6640</td>
<td>700</td>
<td>Daly City Station Platform</td>
<td>36</td>
<td>-0.80%</td>
<td>Open</td>
</tr>
<tr>
<td>Grade</td>
<td>6640</td>
<td>6741</td>
<td>101</td>
<td></td>
<td>36</td>
<td>-0.80%</td>
<td>Open</td>
</tr>
<tr>
<td>Exposure</td>
<td>6741</td>
<td></td>
<td></td>
<td>Gate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6741</td>
<td>7588</td>
<td>847</td>
<td>Switches to Crossover and Spur</td>
<td>36</td>
<td>-0.80%</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>7588</td>
<td>8522</td>
<td>934</td>
<td>Gate</td>
<td>36</td>
<td>-0.80%</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>8522</td>
<td>8722</td>
<td>200</td>
<td>Parabolic grade transition. Midpoint at 8622.</td>
<td>80</td>
<td>transition</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>8722</td>
<td>9003</td>
<td>281</td>
<td></td>
<td>80</td>
<td>2.75%</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>9003</td>
<td></td>
<td></td>
<td>Control Station Boundary</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A train should never get so close to its leading train that if the leading train stops abruptly (e.g., derails) a collision will result.

A train should stop at gates when told (and able) to do so.

A train should not exceed the speed limit of the track segment on which it is travelling.
Secondary Behavioral Requirements

- Optimize throughput.
- Avoid large and frequent changes in speed and acceleration.
- Avoid mode changes. That is, changing from acceleration to braking (or vice versa).
Interesting (aka Fun) Facts

- A train is not a point mass. A train consisting of 10 cars has a total length of 710 feet.
- On a continuous 4% downgrade, at train travelling at 80 mph would take over 3 miles to stop.
Domain Specific Language

- called SPC
- has abstractions suitable for specifying control algorithms
- has a continuous semantics and a discrete semantics

Continuous Framework

- Facilitates a relatively direct formalization of safety properties.

Discrete Framework

- Represents a conservative bounding of the continuous framework.
- Bounds are defined with respect to the system snapshots that are available within a sense-react loop.
Code Generation

- Within the discrete framework, correctness-preserving transformation can be used to derive executable specifications.
- Derivations may also include various optimizations including short-circuiting computations involving inductively defined boolean expressions.
- Optimizing derivations produce implementations that are significantly more efficient than executable specifications.

Correctness

- A number of theorems have been proved (by hand) showing that the discrete framework is consistent with the continuous framework.
- Verification assures that safety properties formalized in the continuous framework have a consistent interpretation in the discrete framework.
A Formal Specification of Safety

\[
\begin{align*}
\text{SafeState}(state_{ot}, state_{lt}, track, signals) & \overset{\text{def}}{=} \\
nStop(state_{ot}) & \ll_S track \land \\
nStop(state_{ot}) & \ll_S signals \land \\
nStop(state_{ot}) & \ll_T \text{currentStop}(state_{lt}) \land \\
eStop(state_{ot}) & \ll_T eStop(state_{lt}) \land \\
eStop(state_{ot}) & \ll_T dStop(state_{lt})
\end{align*}
\]
BART: Raw Data
BART: Visualization of Raw Data