Developing Control Systems in Event-B

Thai Son Hoang

Chair of Information Security, Department of Computer Science
Swiss Federal Institute of Technology Zürich (ETH Zürich)

16th November 2010, McMaster University

Event-B Modelling Method

- A modelling language for discrete transition systems.
- Mathematical language of first-order logic and some typed set theory.
- Incremental development process using refinement.
- Consistency of models: discharging proof obligations.
- Correct-by-construction systems.
- Supported by the RODIN Platform.

Event-B Models

Context
- constants
- carrier sets
- axioms

Machine
- variables: \( v \)
- invariant: \( I(v) \)
- events: \( \text{evt} \)

Static part
- \( t \) – the parameters.
- \( G(t, v) \) – the guard: enable conditions.
- \( v := E(t, v) \) – the action: \( v \) is assigned the value of \( E(t, v) \).
- Initialisation: A special event without parameters and guards.

Dynamic part
- \( \text{evt} \)
- any \( t \) where
- \( G(t, v) \)
- then
- \( v := E(t, v) \)
- end

Consistency: Invariant establishment and preservation
**Refinement**

- A way to introduce more concrete details into the formal model.
- The concrete model must be consistent with the abstract model.
- Analogies with a microscope or a parachute.
- The view of the system gets more accurate.
- Allow to observe the system with a finer time grain.

---

**Applications**

Event-B can be used to model:
- distributed systems,
- concurrent systems,
- sequential programs,
- electronic circuits,
- control systems,
- etc.

---

**Example. File Transfer**

<table>
<thead>
<tr>
<th>Model 0</th>
<th>Model 1</th>
<th>Model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy[d_1, \ldots, d_n]</td>
<td>transfer[d_1]</td>
<td>send[d_1]</td>
</tr>
<tr>
<td></td>
<td>transfer[d_n]</td>
<td>receive[d_1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>send[d_n]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>receive[d_n]</td>
</tr>
</tbody>
</table>

- Model 0: the file is copied in one-shot.
- Model 1: the file is transferred piece-by-piece.
- Model 2: each transfer is done via a pair of send/receive actions.
Event-B Modelling Method

Developing Control Systems
- A Requirements Document
- A Modelling Guideline
- Formal Development

Summary

Train Control at a Stations
- Joint work with Simon Hudon.
- A station has a single track.
- The track is one way:
  - the train enters the station block via the approaching block.
  - the train exits the station block via the exiting block.
- There are two signals located at the two ends of the station.
- The signals turn to red automatically when a train passes by.
- The system controls when to turn the signals to green.

Environment

<table>
<thead>
<tr>
<th>ENV 1</th>
<th>A train occupies no more than one block.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENV 2</td>
<td>Each signal is either green or red.</td>
</tr>
<tr>
<td>ENV 3</td>
<td>Trains are assumed to stop at red signals.</td>
</tr>
<tr>
<td>ENV 4</td>
<td>The signals automatically change from green to red when some train passes by.</td>
</tr>
</tbody>
</table>

Safety Requirement
- The system guarantees that there is no collision between trains.

| SAF 5 | Two trains are not on the same block at the same time. |
Train Schedule

- Each train is associated with a predefined route plan.
- The plan specifies either the train to stop or pass through.

**FUN 6**
Each train either stops or passes through according to a predefined route plan.

Sensors and Actuators

- There are sensors detecting if a block is occupied.
- There are sensors detecting the status of the two signals.

**ENV 7**
The sensors always reflect the values of the corresponding physical components.

**ENV 8**
For each signal, there is an actuator for the controller to turn it from red to green.

A Modelling Guideline for Developing Control Systems

**Phase 1. Environment**
First Model, Trains and Tracks (1/3): The Context

- **carrier sets:** BLOCK, TRAIN
- **constants:** APP, STN, EXT

**axioms:**
axm0_1: partition(BLOCK, {APP}, {STN}, {EXT})

- axm0_1: APP, STN, EXT are distinct blocks.

Phase 1 Model the environment.
Phase 2 Model the actuators.
Phase 3 Model the sensors and the controller.
Phase 4 Model the schedule.
**Phase 1. Environment**

**First Model. Trains and Tracks (2/3): The State**

- **variables:** Trns, Loc, Occ

- **init**
  ```
  begin
  Trns, Loc, Occ := ∅, ∅, ∅
  end
  ```

- **invariants:**
  - `inv0_1`: `Trns` is the set of “monitored” trains.
  - `inv0_2`: Each monitored train occupies one block (ENV 1).
  - `inv0_3`: No two trains are on the same block (SAF 5).
  - `inv0_4`: No two trains are on the same block (SAF 5).

**Phase 1. Environment**

**Second Model. Signals**

- **carrier sets:** LIGHT
- **constants:** RED, GREEN

- **axioms:**
  ```
  axm1_1: partition(LIGHT, {RED}, {GREEN})
  ```

- **variables:** Ent_sgn, Ext_sgn

- **init**
  ```
  begin
  Ent_sgn, Ext_sgn := RED, RED
  end
  ```

- **invariants:**
  - `inv1_1`: `Ent_sgn ∈ LIGHT`
  - `inv1_2`: `Ext_sgn ∈ LIGHT`

  - `ENV 2: Signals are either red or green.`
  - `ENV 3: Trains suppose to obey the signals.`
  - `ENV 4: Signal changes from green to red automatically`.

  - Addition invariant (due to guard strengthening)
    ```
    inv1_3: Ent_sgn = GREEN ⇒ STN ∈ Occ
    ```

**Phase 1. Environment**

**First Model. Trains and Tracks (3/3): The Events**

- There are 4 events modelling the movements of trains.

  - **APPROACH**
    ```
    any t where t /∈ Trns
    APP /∈ Occ
    then
    Trns := Trns ∪ {t}
    Loc(t) := APP
    Occ := Occ ∪ {APP}
    end
    ```

  - **ENTER**
    ```
    any t where t ∈ Trns
    Loc(t) = APP
    STN /∈ Occ
    then
    Loc(t) := STN
    Occ := (Occ \ {STN}) \ {APP}
    end
    ```

  - **guards guarantee safety properties.**
  - **Events EXIT and LEAVE are similar.**
Phase 1. Environment
Second Model. Signals

**CHANGE_ENTER_SIGNAL**

```plaintext
when
  Ent_sgn = RED
  STN /∈ Occ
then
  Ent_sgn := GREEN
end
```

- Recall: invariant inv1_3
  ```plaintext
  inv1_3 : Ent_sgn = GREEN ⇒ STN /∈ Occ
  ```

- Similar for events EXIT and **CHANGE_EXIT_SIGNAL**.

---

Phase 2. Actuators

**variables:** . . . , act_ent_sgn, act_ext_sgn

- **invariants:**
  ```plaintext
  inv2_1 : act_ent_sgn ∈ BOOL
  inv2_2 : act_ext_sgn ∈ BOOL
  ```

- **init**
  ```plaintext
  begin
    . . .
    act_ent_sgn, act_ext_sgn := FALSE, FALSE
  end
  ```

- Additional invariant
  ```plaintext
  inv2_3 : act_ent_sgn = TRUE⇒
  Ent_sgn = RED ∧ STN /∈ Occ
  ```

- **ctrl_change_enter_signal**
  ```plaintext
  when
    act_ent_sgn = FALSE
    Ent_sgn = RED
    STN /∈ Occ
  then
    act_ent_sgn := TRUE
end
```

- Take into account the following invariant
  ```plaintext
  inv2_3 : act_ent_sgn = TRUE⇒
  Ent_sgn = RED ∧ STN /∈ Occ
  ```

- ENV 8: The signals is changed accordingly to the actuators.

- Similar for event **CHANGE_EXIT_SIGNAL**.

---

Thai Son Hoang (ETH-Zürich) | Signal Control using Event-B | McMaster University, 16/11/10 21 / 32
---

Thai Son Hoang (ETH-Zürich) | Signal Control using Event-B | McMaster University, 16/11/10 22 / 32

Thai Son Hoang (ETH-Zürich) | Signal Control using Event-B | McMaster University, 16/11/10 23 / 32

Thai Son Hoang (ETH-Zürich) | Signal Control using Event-B | McMaster University, 16/11/10 24 / 32
Phase 3. Sensors and Controller

variables: ... 
  sen_blk, sen_ent_sgn, sen_ext_sgn

invariants: 
  inv3_1: sen_blk = Occ
  inv3_2: sen_ent_sgn = Ent_sgn
  inv3_3: sen_ext_sgn = Ext_sgn

init
  begin
    ...
    sen_blk := ∅
    sen_ent_sgn, sen_ext_sgn := RED, RED
  end

- \textit{sen blk}: Sensors detecting if a block is occupied.
- \textit{sen ent_sgn, sen ext_sgn}: Sensors detecting status of signals.
- Invariants: Sensors reflect the status of components (ENV 7).

Phase 4. Schedule

- \textit{FUN 6}: Every train has some predefined route plan.
  \begin{itemize}
    \item constants: plan
    \item axioms: \texttt{axm4} \_ \texttt{1}: plan \in TRAIN \rightarrow BOOL
  \end{itemize}

- Plan of the train at the approaching block.
  \begin{itemize}
    \item variables: a_plan
    \item invariants: inv4 \_ \texttt{1}: \quad \forall t \in Tms.
      \texttt{Loc}(t) = APP \Rightarrow a_plan = \texttt{plan}(t)
  \end{itemize}

\begin{enumerate}
  \item Additional assignment(s) in physical events set the value of the sensor appropriately.
  \item Example
  \begin{verbatim}
  ENTER
  any t where
    ...
    then
      Loc := STN
      Occ := (Occ \cup \{STN\}) \setminus \{APP\}
      Ent_sgn := RED
      sen_blk := (sen_blk \cup \{STN\}) \setminus \{APP\}
      sen_ent_sgn := RED
  end
  \end{verbatim}
\end{enumerate}
Schedule appropriately using the plan.

Change the signals only when it is necessary.

Phase 4. Schedule

\[
\begin{align*}
\text{ctrl\_change\_enter\_signal} & \quad \text{when} \quad \ldots \\
& \quad a_{\text{plan}} = \text{TRUE} \\
& \quad \text{APP} \in \text{sen\_blk} \\
& \quad \text{then} \quad \ldots \\
& \quad \text{end} \\
\text{ctrl\_change\_both\_signal} & \quad \text{when} \quad \ldots \\
& \quad a_{\text{plan}} = \text{FALSE} \\
& \quad \text{APP} \in \text{sen\_blk} \\
& \quad \text{then} \quad \ldots \\
& \quad \text{end} \\
\text{ctrl\_change\_exit\_signal} & \quad \text{when} \quad \ldots \\
& \quad \text{STN} \in \text{sen\_blk} \\
& \quad \text{then} \quad \ldots \\
& \quad \text{end}
\end{align*}
\]

Development Summary

<table>
<thead>
<tr>
<th>Phase</th>
<th>Model</th>
<th>Requirement(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>Model 0</td>
<td>ENV 1, SAF 5</td>
</tr>
<tr>
<td></td>
<td>Model 1</td>
<td>ENV 2, ENV 3, ENV 4</td>
</tr>
<tr>
<td>Phase 2</td>
<td>Model 2</td>
<td>ENV 8</td>
</tr>
<tr>
<td>Phase 3</td>
<td>Model 3</td>
<td>ENV 7</td>
</tr>
<tr>
<td>Phase 4</td>
<td>Model 4</td>
<td>FUN 6</td>
</tr>
</tbody>
</table>

Summary. Event-B Modelling Method

- A modelling method for discrete transition systems.
- Mathematical language of first-order logic and set theory.
- Step-wise refinement to reduce development complexity.
- Correct by construction.
- Can be used to model a wide range of applications.

Summary. Developing Control System

- Start with model of the problem: the environment with various constraints.
- Step-by-step introduce:
  - Actuators (output of the controller).
  - Sensors (input of the controller) and the controller.
- Schedule the controller appropriately.
- Important features of the approach:
  - Safety properties are introduced early in terms of the environment: Safety properties are maintained by refinement.
  - Scheduling details in later phase of the development: Separation of concerns between safety properties and schedule.