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

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

INFORMATION SECURITY

Outline

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

Static part

Machine
variables: v
invariant: $I(v)$
events: evt

Dynamic part

```
evt
  any  t  where
    G(t, v)
  then
    v := E(t, v)
  end
```

- 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.

Consistency: Invariant establishment and preservation

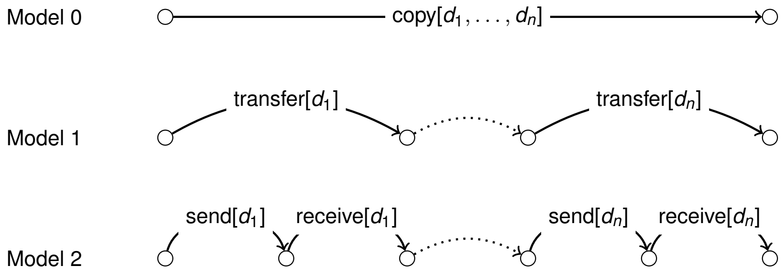
ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

INFORMATION SECURITY

Refinement

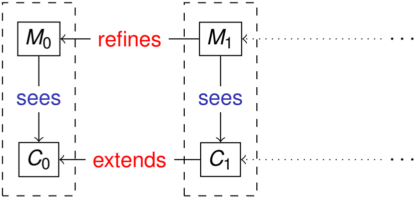
- A way to introduce more concrete details into the formal model.
- The concrete model must be consistent with the abstract model.
- Analogies with a microscope or a parachute.
- The view of the system gets more accurate.
- Allow to observe the system with a finer time grain.

Example. File Transfer



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

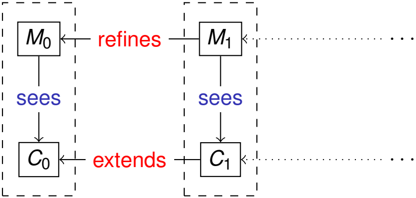
Event-B Refinement



Consistency: The concrete model only exhibits behaviours allowed by the abstract model.

- Event-wise reasoning:
  - Guard strengthening:
    concrete guards are stronger than abstract guards.
  - Simulation: The abstract event can simulate the concrete event.

Event-B Refinement



Consistency: The concrete model only exhibits behaviours allowed by the abstract model.

- Event-wise reasoning:
  - Guard strengthening:
    concrete guards are stronger than abstract guards.
  - Simulation: The abstract event can simulate the concrete event.

Applications

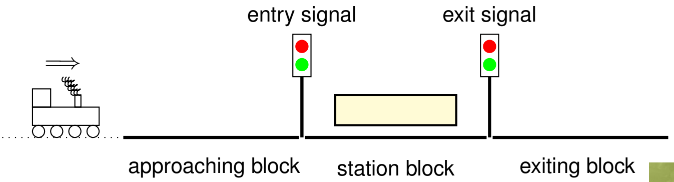
Event-B can be used to model:

- distributed systems,
- concurrent systems,
- sequential programs,
- electronic circuits,
- control systems,
- etc.

Outline

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

ENV 1	A train occupies no more than one block.

ENV 2	Each signal is either green or red.

ENV 3	Trains are assumed to stop at red signals.

ENV 4	The signals automatically change from green to red when some train passes by.

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

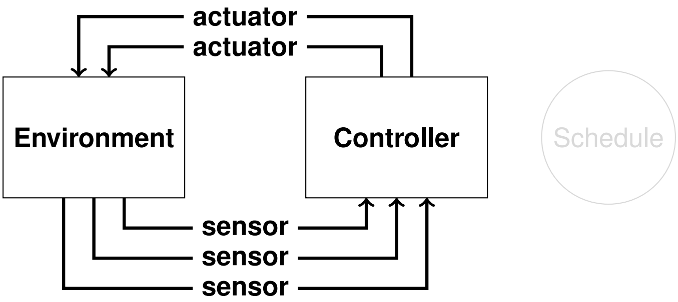
- There are sensors detecting if a block is occupied
- There are sensors detecting the status of the two signals.

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

- The controller commands the signals via actuators.

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



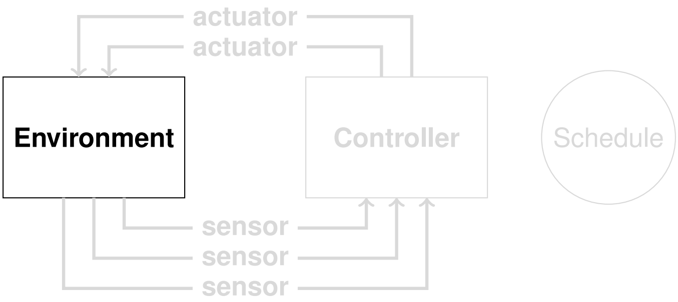
Phase 1 Model the environment.

Phase 2 Model the actuators.

Phase 3 Model the sensors and the controller.

Phase 4 Model the schedule.

A Modelling Guideline for Developing Control Systems



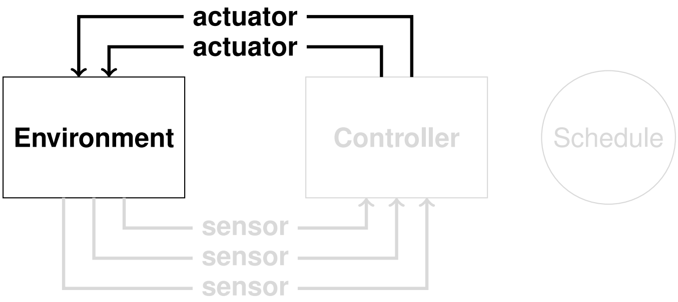
Phase 1 Model the environment.

Phase 2 Model the actuators.

Phase 3 Model the sensors and the controller.

Phase 4 Model the schedule.

A Modelling Guideline for Developing Control Systems



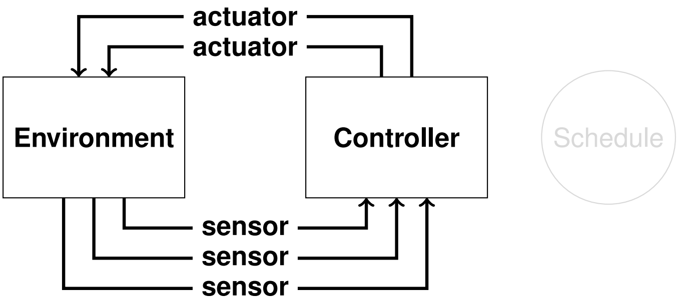
Phase 1 Model the environment.

Phase 2 Model the actuators.

Phase 3 Model the sensors and the controller.

Phase 4 Model the schedule.

A Modelling Guideline for Developing Control Systems



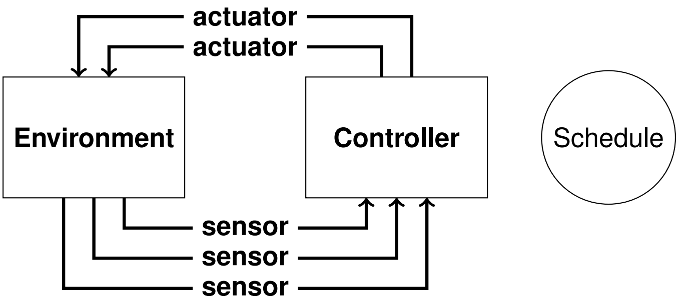
Phase 1 Model the environment.

Phase 2 Model the actuators.

Phase 3 Model the sensors and the controller.

Phase 4 Model the schedule.

A Modelling Guideline for Developing Control Systems



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 (1/3): The Context

carrier sets: $BLOCK, TRAIN$

constants: APP, STN, EXT

axioms:

axm0_1 : $\mathrm{partition}(BLOCK, \{APP\}, \{STN\}, \{EXT\})$

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

Phase 1. Environment

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

variables: $Trns$, Loc, Occ

init
begin
$Trns, Loc, Occ := \varnothing, \varnothing, \varnothing$
end

invariants:

inv0_1 : $Trns \subseteq TRAIN$

inv0_2 : $Loc \in Trns \to BLOCK$

inv0_3 : $Occ = \mathrm{ran}(Loc)$

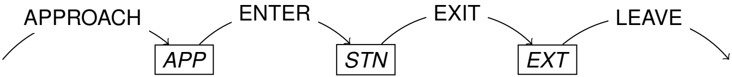
inv0_4 : $\forall t_1, t_2 \cdot t_1 \in Trns \wedge t_2 \in Trains \wedge t_1 \neq t_2 \Rightarrow Loc(t_1) \neq Loc(t_2)$

- **inv0_1**: *Trns* is the set of "monitored" trains.
- **inv0_2–3**: Each monitored train occupies one block (ENV 1).
- **inv0_4**: No two trains are on the same block (SAF 5).

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**
    $t \notin Trns$
    $APP \notin Occ$
  then
    $Trns := Trns \cup \{t\}$
    $Loc(t) := APP$
    $Occ := Occ \cup \{APP\}$
  end

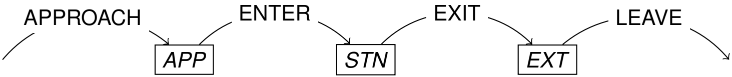
ENTER
  any t **where**
    $t \in Trns$
    $Loc(t) = APP$
    $STN \notin Occ$
  then
    $Loc(t) := STN$
    $Occ := (Occ \cup \{STN\}) \setminus \{APP\}$
  end

- Guards guarantee safety properties.
- Events EXIT and LEAVE are similar.

Phase 1. Environment

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

- There are 4 events modelling the movements of trains.



$$
\begin{array}{l}
\text{APPROACH} \\
\quad \textbf{any} \quad t \quad \textbf{where} \\
\quad\quad t \notin Trns \\
\quad\quad APP \notin Occ \\
\quad \textbf{then} \\
\quad\quad Trns := Trns \cup \{t\} \\
\quad\quad Loc(t) := APP \\
\quad\quad Occ := Occ \cup \{APP\} \\
\quad \textbf{end}
\end{array}
$$

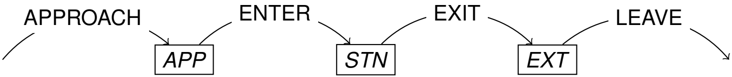
$$
\begin{array}{l}
\text{ENTER} \\
\quad \textbf{any} \quad t \quad \textbf{where} \\
\quad\quad t \in Trns \\
\quad\quad Loc(t) = APP \\
\quad\quad STN \notin Occ \\
\quad \textbf{then} \\
\quad\quad Loc(t) := STN \\
\quad\quad Occ := (Occ \cup \{STN\}) \setminus \{APP\} \\
\quad \textbf{end}
\end{array}
$$

- Guards guarantee safety properties.
- Events EXIT and LEAVE are similar.

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**
$t \notin Trns$
$APP \notin Occ$
then
$Trns := Trns \cup \{t\}$
$Loc(t) := APP$
$Occ := Occ \cup \{APP\}$
end

ENTER
any t **where**
$t \in Trns$
$Loc(t) = APP$
$STN \notin Occ$
then
$Loc(t) := STN$
$Occ := (Occ \cup \{STN\}) \setminus \{APP\}$
end

- Guards guarantee safety properties.
- Events EXIT and LEAVE are similar.

Phase 1. Environment

Second Model. Signals

carrier sets: $LIGHT$

constants: $RED, GREEN$

axioms:
axm1_1 : $\text{partition}(LIGHT, \{RED\}, \{GREEN\})$

variables: $\ldots, Ent\_sgn, Ext\_sgn$

init
begin
$\ldots$
$Ent\_sgn, Ext\_sgn := RED, RED$
end

invariants:
inv1_1 : $Ent\_sgn \in LIGHT$
inv1_2 : $Ext\_sgn \in LIGHT$

- ENV 2: Signals are either red or green.

ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

INFORMATION SECURITY

Phase 1. Environment

Second Model. Signals

(abs_)ENTER
any t **where**
...
$STN \notin Occ$
then
...
end

(cnc_)ENTER
any t **where**
...
$Ent\_sgn = GREEN$
then
...
$Ent\_sgn := RED$
end

- 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 : $Ent\_sgn = GREEN \Rightarrow STN \notin Occ$

Phase 1. Environment

Second Model. Signals

CHANGE_ENTER_SIGNAL
when
$Ent\_sgn = RED$
$STN \notin Occ$
then
$Ent\_sgn := GREEN$
end

- Recall: invariant **inv1_3**

inv1_3 : $Ent\_sgn = GREEN \Rightarrow STN \notin Occ$

- Similar for events EXIT and CHANGE_EXIT_SIGNAL.

Phase 2. Actuators

variables: $\ldots, act\_ent\_sgn, act\_ext\_sgn$

invariants:
inv2_1 : $act\_ent\_sgn \in \mathrm{BOOL}$
inv2_2 : $act\_ext\_sgn \in \mathrm{BOOL}$

init
begin
$\ldots$
$act\_ent\_sgn, act\_ext\_sgn := \mathrm{FALSE}, \mathrm{FALSE}$
end

Phase 2. Actuators

(abs_)CHANGE_ENTER_SIGNAL
when
$Ent\_sgn = RED$
$STN \notin Occ$
then
$Ent\_sgn := GREEN$
end

(cnc_)CHANGE_ENTER_SIGNAL
when
$act\_ent\_sgn = \mathrm{TRUE}$
then
$Ent\_sgn := GREEN$
$act\_ent\_sgn := \mathrm{FALSE}$
end

- Additional invariant

inv2_3 : $act\_ent\_sgn = \mathrm{TRUE} \Rightarrow Ent\_sgn = RED \wedge STN \notin Occ$

- ENV 8: The signals is changed accordingly to the actuators.
- Similar for event CHANGE_EXIT_SIGNAL.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

INFORMATION SECURITY

Phase 2. Actuators

ctrl_change_enter_signal
when
$act\_ent\_sgn = \mathrm{FALSE}$
$Ent\_sgn = RED$
$STN \notin Occ$
then
$act\_ent\_sgn := \mathrm{TRUE}$
end

- Take into account the following invariant

inv2_3 : $act\_ent\_sgn = \mathrm{TRUE} \Rightarrow Ent\_sgn = RED \land STN \notin Occ$

- Similar for events ctrl_change_exit_signal and ctrl_change_both_signal.

Phase 3. Sensors and Controller

variables: $\ldots,$ $sen\_blk,$ $sen\_ent\_sgn,$ 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
$\ldots$
$sen\_blk := \varnothing$
$sen\_ent\_sgn, sen\_ext\_sgn := RED, RED$
end

- *sen_blk*: Sensors detecting if a block is occupied.
- *sen_ent_sgn, sen_ext_sgn*: Sensors detecting status of signals.
- Invariants: Sensors reflect the status of components (ENV 7).

Phase 3. Sensors and Controller

- Additional assignment(s) in physical events set the value of the sensor appropriately.
- Example

ENTER
any t **where**
...
then
$Loc(t) := STN$
$Occ := (Occ \cup \{STN\}) \setminus \{APP\}$
$Ent\_sgn := RED$
$sen\_blk := (sen\_blk \cup \{STN\}) \setminus \{APP\}$
$sen\_ent\_sgn := RED$
end

Phase 3. Sensors and Controller

(abs_)ctrl_change_enter_signal
when
$act\_ent\_sgn = \mathrm{FALSE}$
$Ent\_sgn = RED$
$STN \notin Occ$
then
$act\_ent\_sgn := \mathrm{TRUE}$
end

(cnc_)ctrl_change_enter_signal
when
$act\_ent\_sgn = \mathrm{FALSE}$
$sen\_ent\_sgn = RED$
$STN \notin sen\_blk$
then
$act\_ent\_sgn := \mathrm{TRUE}$
end

- Refinement is trivial with the invariants

inv3_1 : $sen\_blk = Occ$
inv3_2 : $sen\_ent\_sgn = Ent\_sgn$

Phase 4. Schedule

- FUN 6: Every train has some predefined route plan.

constants: *plan*

axioms:
axm4_1 : $plan \in TRAIN \rightarrow \mathrm{BOOL}$

- Plan of the train at the approaching block.

variables: *a_plan*

invariants:
inv4_1 : $\forall t \cdot t \in Trns \wedge$
$Loc(t) = APP \Rightarrow$
$a\_plan = plan(t)$

APPROACH
any t **where**
...
then
...
$a\_plan := plan(t)$
end

Phase 4. Schedule

```
ctrl_change_enter_signal
  when
    ...
    a_plan = TRUE
    APP ∈ sen_blk
  then
    ...
  end
```

```
ctrl_change_both_signal
  when
    ...
    a_plan = FALSE
    APP ∈ sen_blk
  then
    ...
  end
```

```
ctrl_change_exit_signal
  when
    ...
    STN ∈ sen_blk
  then
    ...
  end
```

- Schedule appropriately using the plan.
- Change the signals only when it is necessary.

Phase 4. Schedule

ctrl_change_enter_signal
when
...
$a\_plan = \mathrm{TRUE}$
$APP \in sen\_blk$
then
...
end

ctrl_change_both_signal
when
...
$a\_plan = \mathrm{FALSE}$
$APP \in sen\_blk$
then
...
end

ctrl_change_exit_signal
when
...
$STN \in sen\_blk$
then
...
end

- Schedule appropriately using the plan.
- Change the signals only when it is necessary.

Development Summary

Phase	Model	Requirement(s)
Phase 1	Model 0	ENV 1, SAF 5
	Model 1	ENV 2, ENV 3, ENV 4
Phase 2	Model 2	ENV 8
Phase 3	Model 3	ENV 7
Phase 4	Model 4	FUN 6

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.