A Step-wise Development Method with Progress Concerns

Thai Son Hoang
(joint work with Simon Hudon)

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

InfSec Group Seminar, ETH Zurich
26th March 2013
To develop a system $M$ satisfying property $\phi$, i.e., $M \models \phi$

- $M$: some transition system
- $\phi$: some logical formula

The main challenge: the complexity of the system.

Refinement allows the step-by-step design of the system.
**Refinement**

The UNITY way vs. the Event-B way

- **UNITY**: Refines the *formulae*.

  \[
  \phi \leftarrow \phi_1 \leftarrow \ldots \leftarrow \phi_n \leftarrow M
  \]

  Refinement

  Translation

  Cons: Hard to understand the choice of refinement.

- **Event-B**: Refines *transition systems*.

  \[
  \phi \models \begin{array}{c} M_0 \subseteq M_1 \ldots \subseteq M \end{array}
  \]

  Refinement

  Verification

  Cons: No support for liveness properties.
Refinement
The UNITY way vs. the Event-B way

- **UNITY:** Refines the *formulae*.

\[
\phi \leftarrow \phi_1 \leftarrow \ldots \leftarrow \phi_n \models M
\]

Refinement
Translation

- **Cons:** Hard to understand the choice of refinement.

- **Event-B:** Refines *transition systems*.

\[
\phi \models M_0 \sqsubseteq M_1 \ldots \sqsubseteq M
\]

Refinement
Verification

- **Cons:** No support for *liveness properties*.
Inspired by UNITY and Event-B.

Support the reasoning of liveness properties (UNITY).

Refinement of transition systems (Event-B style).

Developments using Unit-B are guided by both safety and liveness requirements.
Outline

1. Formal Systems Development using Refinement

2. The Unit-B Modelling Method
   - Unit-B Models
   - Properties of Unit-B Models
   - Refinement

3. Summary
Running Example. A Signal Control System

SAF 1  There is at most one train on each block

LIVE 2  Each train in the network eventually leaves
Outline

1. Formal Systems Development using Refinement

2. The Unit-B Modelling Method
   - Unit-B Models
   - Properties of Unit-B Models
   - Refinement

3. Summary
States are captured by variables $v$.

Transitions are modelled by guarded and scheduled events.
A trace $\sigma$ is a (finite or infinite sequence of states)

$$\sigma = s_0, s_1, s_2, s_3, \ldots$$

- A (basic) state formula $P$ is any first-order logic formula,

- The basic formulas can be extended by combining the Boolean operators ($\neg$, $\land$, $\lor$, $\Rightarrow$) with temporal operators:

  - **always:** $\square \phi$

  - $\square$ always:

    - $s_0$: $\phi$
    - $s_1$: $\phi$
    - $s_2$: $\phi$
    - $s_3$: $\phi$

  - **eventually:** $\Diamond \phi$

    - $\Diamond$ eventually:

    - $s_0$: $\neg\phi$
    - $s_1$: $\neg\phi$
    - $s_2$: $\phi$
    - $s_3$: $\neg\phi$
Guarded events

\[ e \quad \text{any} \quad t \quad \text{where} \quad G.t.v \quad \text{then} \quad S.t.v.v' \quad \text{end} \]

- \( t \): parameters
- \( G.t.v \): guard
- \( S.t.v.v' \): action

- \( e.t \) is enabled when \( G.t.v \) holds.
- Execution of \( e.t \): \( v \) is updated according to the action \( S.t.v.v' \).
- \( e.t \) corresponds to a formula \( \text{act.}(e.t) \).
Scheduled events (1/2)

\( \exists e \) any \( t \) where

\[
\begin{align*}
\text{during} & \quad C.t.v \\
\text{upon} & \quad F.t.v \\
\text{then} &
\end{align*}
\]

\( \ldots \)

end

- \( C.t.v \): coarse-schedule.
- \( F.t.v \): fine-schedule.
- Healthiness condition:

\[
C.t.v \land F.t.v \Rightarrow G.t.v
\]

Liveness (Scheduling) Assumption

If \( C.t.v \) holds infinitely long and \( F.t.v \) holds infinitely often then eventually \( e.t \) is executed.

\[
sched.(e.t) = \Box (\Box C \land \Box \Diamond F \Rightarrow \Diamond (F \land \text{act.}(e.t)))
\]
Scheduled events (1/2)

\[ e \]

\[
\begin{align*}
&\text{any } t \text{ where} \\
&\text{...}
\end{align*}
\]

\[
\begin{align*}
&\text{during} \\
&C.t.v
\end{align*}
\]

\[
\begin{align*}
&\text{upon} \\
&F.t.v
\end{align*}
\]

\[
\begin{align*}
&\text{then} \\
&\text{...}
\end{align*}
\]

\[
\begin{align*}
&\text{end}
\end{align*}
\]

- \( C.t.v \): coarse-schedule.
- \( F.t.v \): fine-schedule.
- Healthiness condition:
  \[ C.t.v \land F.t.v \Rightarrow G.t.v \]

Liveness (Scheduling) Assumption

If \( C.t.v \) holds infinitely long and \( F.t.v \) holds infinitely often then eventually \( e.t \) is executed.

\[
\text{sched.}(e.t) = \square(\square C \land \square \Diamond F \Rightarrow \Diamond(F \land \text{act.}(e.t)))
\]
Schedules vs. Fairness

\[ e \equiv \text{any } t \text{ where } G.t.v \text{ during } C.t.v \text{ upon } F.t.v \text{ then } \ldots \text{ end} \]

- Schedules are a generalisation of weak- and strong-fairness.

- **Weak-fairness:**
  - If \( e \) is enabled infinitely long then \( e \) eventually occurs.
  - Let \( C \) be \( G \) and \( F \) be \( \top \).

- **Strong-fairness:**
  - If \( e \) is enabled infinitely often then \( e \) eventually occurs.
  - Let \( F \) be \( G \) and \( C \) be \( \top \).
Scheduled events (2/2)

Conventions

\[ e \triangleq \text{any } t \text{ where } \ldots \text{ during } C.t.v \text{ upon } F.t.v \text{ then } \ldots \text{ end} \]

- **Unscheduled events** (without **during** and **upon**): \( C \) is \( \bot \)
- When only **during** is present (no **upon**), \( F \) is \( \top \).
- When only **upon** is present (no **during**), \( C \) is \( \top \).
Execution of Unit-B Models

\[
\begin{align*}
\text{ex.} \, M & = \text{saf.} \, M \land \text{live.} \, M \\
\text{saf.} \, M & = \text{init} \land \Box \, \text{step.} \, M \\
\text{step.} \, M & = (\exists e.t \in M \cdot \text{act.}(e.t)) \lor \text{SKIP} \\
\text{live.} \, M & = \forall e.t \in M \cdot \text{sched.}(e.t) \\
\text{sched.}(e.t) & = \Box(\Box C \land \Box \Diamond F \Rightarrow \Diamond(F \land \text{act.}(e.t)))
\end{align*}
\]
A Signal Control System (Recall)

SAF 1  There is at most one train on each block

LIVE 2  Each train in the network eventually leaves

Refinement strategy: Prioritise LIVE 2 first.
Focus on trains in the network

Set $TRN$ denotes the set of possible trains.

Variable $trns$ denotes the set of trains in the network.

Event $arrive$ models a train entering the network.

Event $depart$ models a train leaving the network.

\[
\begin{align*}
arrive & \quad \text{any } t \quad \text{where} \\
& \quad t \in TRN \\
\text{then} & \quad trns := trns \cup \{t\} \\
\text{end} \\
\end{align*}
\]

\[
\begin{align*}
depart & \quad \text{any } t \quad \text{where} \\
& \quad t \in TRN \\
\text{during} & \quad t \in trns \\
\text{then} & \quad trns := trns \setminus \{t\} \\
\text{end} \\
\end{align*}
\]
Outline

1. Formal Systems Development using Refinement

2. The Unit-B Modelling Method
   - Unit-B Models
   - Properties of Unit-B Models
   - Refinement

3. Summary
Execution and Properties

\[ \mathcal{M} \text{ satisfies } \phi \text{ if and only if } \exists \mathcal{M} \Rightarrow \phi. \]
Safety Properties

- **Invariance** properties: (in LTL $\square P$)
  - $P$ holds for every reachable state.
  - Proved using the standard induction technique.

- **Unless** properties: $P \text{ un } Q$
  - if $P$ holds at some point then it continues to hold unless $Q$ holds.
  - Prove: If for every event $e \equiv \text{any } t \text{ where } G.t.v \text{ during } \ldots \text{ upon } \ldots \text{ then } S.t.v.v' \text{ end}$ in $M$, we have
    
    $$P.v \land \neg Q.v \land G.t.v \land S.t.v.v' \Rightarrow P.v' \lor Q.v'$$

    then $M$ satisfies $P \text{ un } Q$. 

Liveness Properties

- **Progress properties** $P \leadsto Q$.
- In LTL: $\square (P \Rightarrow \diamond Q)$
- Some important rules

  $$(P \Rightarrow Q) \Rightarrow (P \leadsto Q)$$  \hspace{2cm} \text{(Implication)}

  $$(P \leadsto Q) \land (Q \leadsto R) \Rightarrow (P \leadsto R)$$  \hspace{2cm} \text{(Transitivity)}

  $$(P \leadsto Q) \iff (P \land \neg Q \leadsto Q)$$  \hspace{2cm} \text{(Split-Off-Skip)}$$
Each train in the network eventually leaves

\[ \text{properties} : \]
\[ \text{prg0_1} : \quad t \in \text{trns} \leadsto t \notin \text{trns} \]

Note: Free-variables are universally quantified.
Transient Properties (1/3)

Definition

- Borrowed from UNITY.
- The basic tool for reasoning about progress properties.
- \( \text{tr } P \) states that always \( P \) is eventually falsified.
- In LTL: \( \Box \Diamond \neg P \).
- Important properties:

\[
\text{tr } P = \top \iff \neg P = P \iff \neg P
\]
Theorem (Implementing tr)

if there exists an event

\[ e \triangleq \text{any } t \text{ where } G \cdot t \cdot v \text{ during } C \cdot t \cdot v \text{ upon } F \cdot t \cdot v \text{ then } S \cdot t \cdot v \cdot v' \text{ end} \]

in \( M \) such that

\[ \Box (P \Rightarrow C) , \]

\[ C \leadsto F , \]

\[ P \cdot v \land C \cdot t \cdot v \land F \cdot t \cdot v \land G \cdot t \cdot v \land S \cdot t \cdot v \cdot v' \Rightarrow \neg P \cdot v' \]

then \( M \) satisfies \( tr \; P \).

- (SCH) corresponds to an invariance property.
- (PRG) is trivial when \( F \) is \( \top \).
- (NEG) corresponds to a standard Hoare-triple.
Consider $\text{tr } P = P \bowtie \neg P = \Box(P \implies \Diamond \neg P)$.

Proof (Sketch).

Assume $P$ holds in some state, we prove $\Diamond \neg P$ by contradiction.

1. Assume $\Box P$.

2. From (SCH), we have $\Box C$,

3. together with (PRG), we have $\Box \Diamond F$.

4. Scheduling assumption ensures that $e$ will eventually occur.

5. (NEG) guarantees that when $e$ occurs, $P$ is falsified.

6. We have a contradiction with the assumption from Step 1.
A Signal Control System. The Initial Model

Properties

\[
\begin{align*}
\text{depart} & \quad \text{any } t \text{ where } t \in \text{TRN} \\
\text{during} & \quad t \in \text{trns} \\
\text{then} & \quad \text{trns} := \text{trns} \setminus \{t\} \\
\text{end} & \quad \text{prg0}_1 : \quad t \in \text{trns} \implies t \notin \text{trns}
\end{align*}
\]

- \text{prg0}_1 \text{ is the same as } \text{tr } t \in \text{trns}
- (SCH) is trivial.
- No fine-schedule (\( F \) is \( \top \)) hence (PRG) is trivial.
- The event falsifies \( t \in \text{trns} \) (NEG)
Outline

1. Formal Systems Development using Refinement

2. The Unit-B Modelling Method
   - Unit-B Models
   - Properties of Unit-B Models
   - Refinement

3. Summary
Refinement

- Abstract systems can simulate behaviours of concrete systems.

\[ \text{ex.cncM} \Rightarrow \text{ex.absM} \]

- Event-based reasoning.

\[
\begin{align*}
\text{(abs\_)}e & \equiv \text{any } t \text{ where } G \text{ during } C \text{ upon } F \text{ then } S \text{ end} \\
\text{(cnc\_)}f & \equiv \text{any } t \text{ where } H \text{ during } D \text{ upon } E \text{ then } R \text{ end}
\end{align*}
\]

- Safety:
  - Guard strengthening: \( H \Rightarrow G \)
  - Action strengthening: \( R \Rightarrow S \)

- Liveness:
  - Liveness assumption strengthening.
  - Schedules weakening:

\[
(\Box C \land \Diamond F) \Rightarrow (\Box D \land \Diamond E)
\]
Schedules Weakening

Practical Rules

\[(\Box C \land \Box \Diamond F) \Rightarrow (\Box D \land \Box \Diamond E)\]  
(REF_LIVE)

- **Practical rules:**
  - Coarse-schedule following
    \[C \land F \leadsto D\]  
    (C_FLW)
  - Coarse-schedule stabilising
    \[D \text{ un } C\]  
    (C_STB)
  - Fine-schedule following
    \[C \land F \leadsto E\]  
    (F_FLW)
Schedules Weakening

Practical Rules

\[(\square C \land \square \Diamond F) \Rightarrow (\square D \land \square \Diamond E)\] (REF_LIVE)

- **Practical rules:**
  - Coarse-schedule following
    \[C \land F \leadsto D\] (C_FLW)
  - Coarse-schedule stabilising
    \[D \underline{\text{un}} C\] (C_STB)
  - Fine-schedule following
    \[C \land F \leadsto E\] (F_FLW)
Introduce the network topology: \textit{BLK}, \textit{Entry}, \textit{PLF}, \textit{Exit}.

Variable $loc$ denotes location of trains in the network.

$\text{inv1}_1 : loc \in \text{trns} \rightarrow \text{BLK}$
A Signal Control System. The First Refinement

Refinement of `depart`

(abs_)depart

any $t$ where $t \in TRN$
during $t \in trns$
then
$$trns := trns \setminus \{t\}$$
end

(cnc_)depart

any $t$ where $t \in trns \land loc.t = Exit$
during $t \in trns \land loc.t = Exit$
then
$$trns := trns \setminus \{t\}$$
$$loc := \{t\} \triangleleft loc$$
end

- Guard and action strengthening are trivial.
- Coarse-schedule following (amongst others):

  $$t \in trns \iff t \in trns \land loc.t = Exit$$  \hspace{1cm} (prg1_1)
A Signal Control System. The First Refinement

Refinement of depart

\[ t \in \text{trns} \implies t \in \text{trns} \land \text{loc}.t = \text{Exit} \]

\[ \iff \quad \text{Put the negation of RHS in the LHS} \]

\[ t \in \text{trns} \land \text{loc}.t \neq \text{Exit} \implies t \in \text{trns} \land \text{loc}.t = \text{Exit} \]

\[ \iff \quad \text{Transitivity} \]

\[ t \in \text{trns} \land \text{loc}.t \neq \text{Exit} \implies t \in \text{trns} \land \text{loc}.t \in \text{PLF} \land \]

\[ t \in \text{trns} \land \text{loc}.t \in \text{PLF} \implies t \in \text{trns} \land \text{loc}.t = \text{Exit} \]

- The 1st condition is implemented by event movein (not shown)
- The 2nd condition is implemented by event moveout
- We need the ensure rule (next slide).
The Ensure Rule

Theorem (The ensure-rule)

For all state predicates $p$ and $q$,

$$(P \text{ un } Q) \land (\text{tr } P \land \neg Q) \Rightarrow (P \leadsto Q)$$

(ENS)
The Ensure Rule

Theorem (The ensure-rule)

For all state predicates $p$ and $q$,

\[(P \operatorname{un} Q) \land (\operatorname{tr} P \land \neg Q) \Rightarrow (P \leadsto Q)\]  

\[(\text{ENS})\]
The Ensure Rule

Theorem (The ensure-rule)

For all state predicates $p$ and $q$,

$$(P \mathbf{un} Q) \land (\mathbf{tr} P \land \neg Q) \Rightarrow (P \rightsquigarrow Q) \quad \text{(ENS)}$$

Diagram:

- $P$
- $Q$
- $P \land \neg Q$
- $P \lor Q$
- $\neg P \lor Q$
- $P \land \neg Q$
- $P \lor Q$
- $P \land \neg Q$
- $P \lor Q$
The Ensure Rule

Theorem (The ensure-rule)

For all state predicates $p$ and $q$,

\[(P \cup Q) \land (\text{tr } P \land \neg Q) \Rightarrow (P \leadsto Q)\]  

(ENS)
The Ensure Rule

Theorem (The ensure-rule)

For all state predicates \( p \) and \( q \),

\[
(P \text{ un } Q) \land (\text{tr } P \land \neg Q) \implies (P \leadsto Q) \tag{ENS}
\]
A Signal Control System. The First Refinement

New Event: moveout

\[
\begin{align*}
t & \in \text{trns} \land \text{loc}.t \in \text{PLF} \quad \sim \quad t \in \text{trns} \land \text{loc}.t = \text{Exit} \\
\iff \\
(t \in \text{trns} \land \text{loc}.t \in \text{PLF}) & \cup \\
(\text{tr} (t \in \text{trns} \land \text{loc}.t \in \text{PLF}) \land \neg (t \in \text{trns} \land \text{loc}.t = \text{Exit})) \\
\end{align*}
\]

Ensure rule

Logic

moveout

any \quad t \quad \text{where}
\begin{align*}
t & \in \text{trns} \land \text{loc}.t \in \text{PLF} \\
during \\
(t \in \text{trns} \land \text{loc}.t \in \text{PLF} \\
then \\
\text{loc}.t & := \text{Exit} \\
end
\end{align*}
A Signal Control System. The First Refinement

New Event \textit{moveout}

\[ t \in \text{trns} \land \text{loc}.t \in \text{PLF} \iff t \in \text{trns} \land \text{loc}.t = \text{Exit} \]

\[
\iff t \in \text{trns} \land \text{loc}.t \in \text{PLF} \quad \text{un} \quad t \in \text{trns} \land \text{loc}.t = \text{Exit} \land \\
\left( \text{tr} \left( t \in \text{trns} \land \text{loc}.t \in \text{PLF} \right) \land \neg \left( t \in \text{trns} \land \text{loc}.t = \text{Exit} \right) \right) \\
\iff \ldots \land \left( \text{tr} t \in \text{trns} \land \text{loc}.t \in \text{PLF} \right) \\
\]

\text{moveout}

\text{any} \quad t \quad \text{where} \quad t \in \text{trns} \land \text{loc}.t \in \text{PLF}

\text{during} \quad t \in \text{trns} \land \text{loc}.t \in \text{PLF}

\text{then} \quad \text{loc}.t := \text{Exit}

\text{end}
A Signal Control System. The Second Refinement

The State

\[ \forall t_1, t_2 \cdot t_1 \in trns \land t_2 \in trns \land \text{loc}.t_1 = \text{loc}.t_2 \Rightarrow t_1 = t_2 \]

**SAF 1** There is at most one train on each block
A Signal Control System. The Second Refinement

Refinement of moveout

(abs_)moveout

any t where
  t ∈ trns ∧ loc.t ∈ PLF
during
  t ∈ trns ∧ loc.t ∈ PLF
then
  loc.t := Exit
end

(cnc_)moveout

any t where
  t ∈ trns ∧ loc.t ∈ PLF ∧ Exit ∉ ran.loc
during
  t ∈ trns ∧ loc.t ∈ PLF
upon
  Exit ∉ ran.loc
then
  loc.t := Exit
end

Neither weak- nor strong-fairness is satisfactory.

- Weak-fairness requires Exit to be free infinitely long.
- Strong-fairness is too strong assumption.
A Signal Control System. The Second Refinement

Refinement of \textit{moveout}

\begin{align*}
\text{(abs\_)}\text{moveout} & \quad \text{any } t \text{ where } \\
& \quad t \in \text{trns} \land \text{loc} \cdot t \in \text{PLF} \quad \text{during} \\
& \quad t \in \text{trns} \land \text{loc} \cdot t \in \text{PLF} \quad \text{then} \\
& \quad \text{loc} \cdot t := \text{Exit} \\
\text{end} \\
\end{align*}

\begin{align*}
\text{(cnc\_)}\text{moveout} & \quad \text{any } t \text{ where } \\
& \quad t \in \text{trns} \land \text{loc} \cdot t \in \text{PLF} \land \text{Exit} \notin \text{ran} \cdot \text{loc} \quad \text{during} \\
& \quad t \in \text{trns} \land \text{loc} \cdot t \in \text{PLF} \quad \text{upon} \\
& \quad \text{Exit} \notin \text{ran} \cdot \text{loc} \quad \text{then} \\
& \quad \text{loc} \cdot t := \text{Exit} \\
\text{end} \\
\end{align*}

- Neither weak- nor strong-fairness is satisfactory.
  - Weak-fairness requires \textit{Exit} to be free infinitely long.
  - Strong-fairness is too strong assumption.
A Signal Control System. The Second Refinement

Refinement of \texttt{moveout}

\begin{verbatim}
(abs_)moveout
  any \ t \ where
    t \in trns \land loc.t \in PLF
during
    t \in trns \land loc.t \in PLF
then
  loc.t := Exit
end
\end{verbatim}

\begin{verbatim}
(cnc_)moveout
  any \ t \ where
    t \in trns \land loc.t \in PLF
  during
    t \in trns \land loc.t \in PLF
  upon
    Exit \not\in \text{ran}.loc
then
  loc.t := Exit
end
\end{verbatim}

- Neither weak- nor strong-fairness is satisfactory.
  - Weak-fairness requires \textit{Exit} to be free infinitely long.
  - Strong-fairness is too strong assumption.
Introduce the signals \( sgn \)

\[
\begin{align*}
\text{inv3}_1 & : \quad sgn \in \{ \text{Entry} \} \cup \text{PLF} \to \text{COLOR} \\
\text{inv3}_2 & : \quad \forall p \cdot p \in \text{PLF} \land sgn.p = \text{GR} \Rightarrow \text{Exit} \notin \text{ran} \cdot \text{loc} \\
\text{inv3}_3 & : \quad \forall p, q \cdot p, q \in \text{PLF} \land sgn.p = sgn.q = \text{GR} \Rightarrow p = q
\end{align*}
\]
A Signal Control System. The Third Refinement

Refinement of moveout

(abs_)moveout

any t where
  t ∈ trns ∧ loc.t ∈ PLF ∧
  Exit ∉ ran loc
during
  t ∈ trns ∧ loc.t ∈ PLF
upon
  Exit ∉ ran loc
then
  loc.t := Exit
end

(cnc_)moveout

any t where
  t ∈ trns ∧ loc.t ∈ PLF ∧
  sgn.(loc.t) = GR
during
  t ∈ trns ∧ loc.train ∈ PLF ∧
  sgn.(loc.t) = GR
then
  loc.t := Exit
  sgn.(loc.t) := RD
end

Refinement requires to prove:

\[ tr \, t \in \, trns \land loc.t \in \, PLF \land sgn.(loc.t) = RD. \] (prg3_5)
A Signal Control System. The Third Refinement

New Controller Event `ctrl_platform`

```plaintext
ctrl_platform
    any  p  where
    p ∈ PLF ∧ p ∈ ran loc ∧ Exit /∈ ran loc ∧
    (∀q·q ∈ PLF ⇒ sgn.q = RD)
  during
  p ∈ PLF ∧ p ∈ ran loc ∧ sgn.p = RD
  upon
  Exit /∈ ran(loc) ∧ (∀q·q ∈ PLF ∧ q ≠ p ⇒ sgn.q = RD)
  then
  sgn.p := GR
  end
```
Summary
The Unit-B Modelling Method

- Guarded and scheduled events.
- Reasoning about liveness (progress) properties.
- Refinement preserving safety and liveness properties.
- Developments are guided by safety and liveness requirements.
Summary

Future Work

- Data refinement
- Decomposition / Composition
- Tool support
References I

Simon Hudon.
A Progress Preserving Refinement.
*Master Thesis.*
Chair of Information Security, ETH Zurich, 2011.

Simon Hudon and Thai Son Hoang.
Systems Design Guided by Progress Concerns.
Accepted for *iFM 2013.*