Progress Concerns as Design Guidelines

Simon Hudon[1] and Thai Son Hoang[2]

[1]Department of Computer Science, York University, Canada
[2]Institute of Information Security, ETH Zurich, Switzerland

iFM 2013, Turku, Finland
12th June 2013

Safety vs. Liveness

Liveness Properties

- Something (good) will happen
- e.g. termination, progress

Safety Properties

- Something (bad) never happens.
- e.g. invariance properties

Safety vs. Liveness

Liveness Properties

- Something (good) will happen
- e.g. termination, progress

Safety Properties

- Something (bad) never happens.
- e.g. invariance properties

- Liveness properties are desirable.

Safety vs. Liveness

Liveness Properties

- Something (good)
  will happen
- e.g. termination, progress

Safety Properties

- Something (bad)
  never happens.
- e.g. invariance properties

- Liveness properties are desirable.

Safety vs. Liveness

Liveness Properties

- Something (good)
  will happen
- e.g. termination, progress

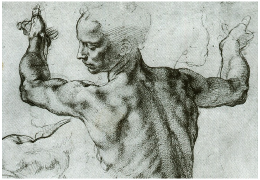
Safety Properties

- Something (bad)
  never happens.
- e.g. invariance properties

- Liveness properties are desirable.

- Event-B does not support liveness properties.

Systems Development using Event-B



$\boxed{\mathbf{M}_0}$ —— refined by $\longrightarrow$ $\boxed{\mathbf{M}_1}$ ············ refined by ············$\rightarrow$ $\boxed{\mathbf{M}_n}$

satisfies satisfies satisfies

ϕ_0 ϕ_1 ϕ_n

- $\phi_0, \phi_1, \ldots, \phi_n$: safety properties.

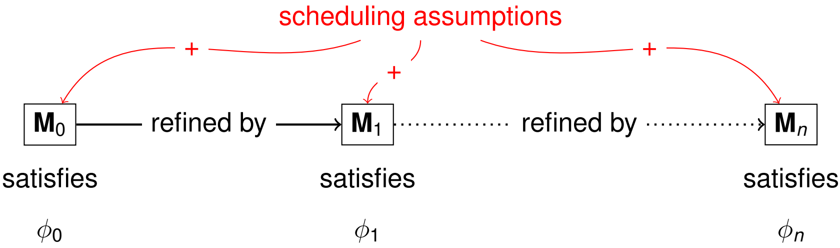
Unit-B = UNITY + Event-B

$\boxed{\mathbf{M}_0}$ —— refined by ⟶ $\boxed{\mathbf{M}_1}$ ⋯⋯ refined by ⋯⋯→ $\boxed{\mathbf{M}_n}$

satisfies	satisfies	satisfies
ϕ_0	ϕ_1	ϕ_n

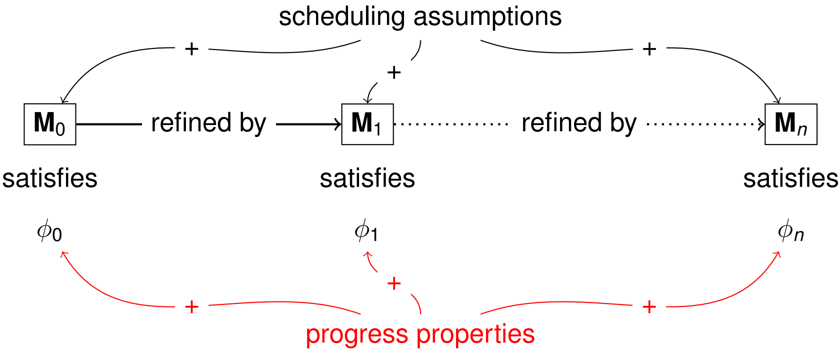
- Developments using Unit-B are
  guided by both safety and liveness requirements.

Unit-B = UNITY + Event-B



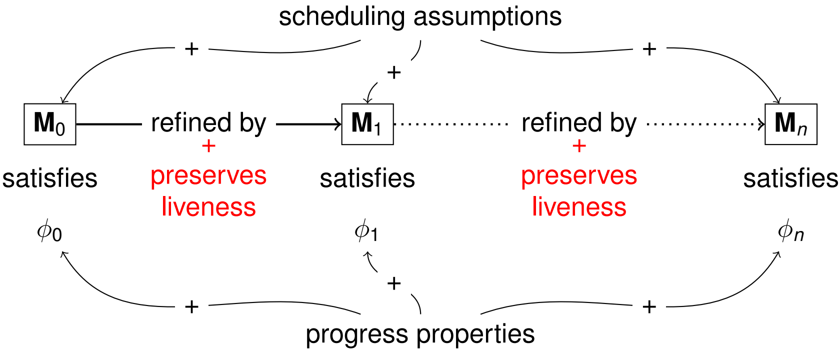
- Developments using Unit-B are
  guided by both safety and liveness requirements.

Unit-B = UNITY + Event-B



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

Unit-B = UNITY + Event-B



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

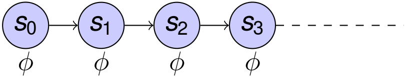
Traces and the Language of Temporal Logic

A trace σ is a (finite or infinite sequence of states)

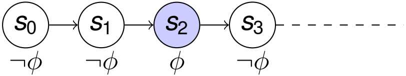
$$\sigma = s_0, s_1, s_2, s_3, \ldots$$

- A (basic) state formula P is any first-order logic formula,
- The basic formulas can be extended by combining the Boolean operators $(\neg, \wedge, \vee, \Rightarrow)$ with temporal operators:

always: $\square\, \phi$



eventually: $\lozenge\, \phi$



Unit-B Models. Guarded and Scheduled Events

e
  any t **where**
    $G.t.v$
  during
    $C.t.v$
  upon
    $F.t.v$
  then
    $S.t.v.v'$
  end

- Execution of $\mathrm{e}.t$ corresponds to a formula $act.(\mathrm{e}.t)$.
- $C.t.v$: coarse-schedule.
- $F.t.v$: fine-schedule.

Liveness (Scheduling) Assumption

If $C.t.v$ holds infinitely long and $F.t.v$ holds infinitely often then eventually $\mathrm{e}.t$ is executed.

$$sched.(\mathrm{e}.t) \;=\; \Box(\Box\, C \wedge \Box \Diamond\, F \;\Rightarrow\; \Diamond(act.(\mathrm{e}.t)))$$

Unit-B Models. Guarded and Scheduled Events

e
  any t **where**
    $G.t.v$
  during
    $C.t.v$
  upon
    $F.t.v$
  then
    $S.t.v.v'$
  end

- Execution of e.t corresponds to a formula $act.(\mathrm{e}.t)$.
- $C.t.v$: 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.

$$sched.(\mathrm{e}.t) \;=\; \Box(\Box\, C \land \Box\Diamond\, F \;\Rightarrow\; \Diamond(act.(\mathrm{e}.t)))$$

Schedules vs. Fairness

$e \mathrel{\widehat{=}}$ **any** t **where** $G.t.v$ **during** $C.t.v$ **upon** $F.t.v$ **then** $\ldots$ **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

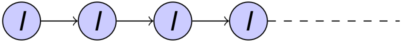
e $\widehat{=}$ **any** t **where** ... **during** $C.t.v$ **upon** $F.t.v$ **then** ... **end**

- Unscheduled events (without **during** and **upon**): C is $\perp$
- When only **during** is present (no **upon**), F is $\top$.
- When only **upon** is present (no **during**), C is $\top$.

Safety Properties

- Invariance properties: (in LTL $\Box\, I$)

$$\Box\, I$$

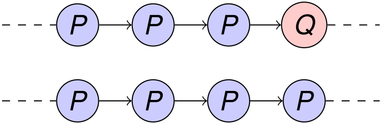


- Unless properties: $P\ \mathbf{un}\ Q$

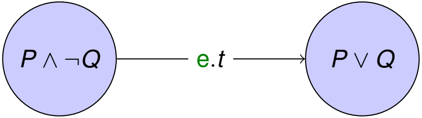
$$P \Rightarrow P\,\mathcal{U}\,Q$$

or

$$P \Rightarrow \Box\, P$$



  - Prove: For every event e.t in **M**



Liveness Properties

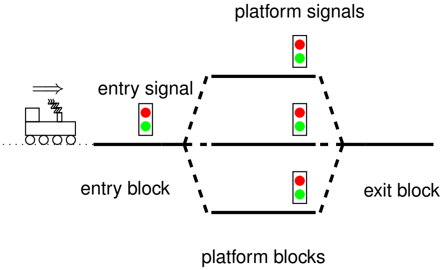
- Progress properties

$$P \leadsto Q \;\;\widehat{=}\;\; \Box(P \Rightarrow \Diamond Q)$$

- Some important rules

$$\begin{array}{rcll} (P \Rightarrow Q) & \Rightarrow & (P \leadsto Q) & \text{(Implication)} \\ (P \leadsto Q) \wedge (Q \leadsto R) & \Rightarrow & (P \leadsto R) & \text{(Transitivity)} \end{array}$$

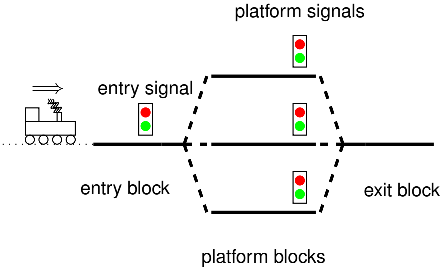
A Signal Control System



SAF 1 There is at most one train on each block

LIVE 2 Each train in the network eventually leaves

A Signal Control System



SAF 3 There is at most one train on each block

LIVE 4 Each train in the network eventually leaves

Refinement Strategy

Model 0 To model trains in the network, focus on LIVE 2

Ref. 1 To introduce the network topology

Ref. 2 To take into account SAF 1

Ref. 3 To introduce signals and derive a spec for the controller

A Signal Control System. The Initial Model

Sketch

LIVE 2 Each train in the network eventually leaves

variables : $trns$

invariants :
$trns \subseteq TRN$

A Signal Control System. The Initial Model

Sketch

LIVE 2 Each train in the network eventually leaves

variables : *trns*

invariants :
$trns \subseteq TRN$

arrive
  any t **where**
    $t \in TRN$
  then
    $trns := trns \cup \{t\}$
  end

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

A Signal Control System. The Initial Model

Sketch

LIVE 2 Each train in the network eventually leaves

variables : $trns$

invariants :
$trns \subseteq TRN$

arrive
any t **where**
$t \in TRN$
then
$trns := trns \cup \{t\}$
end

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

properties :
prg0_1 : $t \in trns \leadsto t \notin trns$

Note: Free variables are universally quantified.

Transient Properties

Theorem (Implementing $P \leadsto \neg P$)

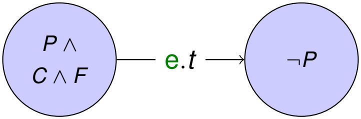
M *satisfies* $P \leadsto \neg P$ *if there exists an event in* **M**

$$\mathsf{e} \mathrel{\widehat{=}} \textbf{any } t \textbf{ where } G.t.v \textbf{ during } C.t.v \textbf{ upon } F.t.v \textbf{ then } S.t.v.v' \textbf{ end}$$

such that

$$\Box(P \Rightarrow C)\ , \qquad \text{(SCH)}$$

$$C \leadsto F\ , \qquad \text{(PRG)}$$



(NEG)

A Signal Control System. The Initial Model

Properties

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

prg0_1 : $t \in trns \leadsto t \notin trns$

- (SCH) is trivial.
- No fine-schedule (F is $\top$) hence (PRG) is trivial.
- The event falsifies $t \in trns$ (NEG)

Refinement

- Abstract systems can simulate behaviours of concrete systems.

$$ex.\mathbf{cncM} \;\Rightarrow\; ex.\mathbf{absM}$$

- Event-based reasoning.

$$(\mathrm{abs\_})\mathrm{e} \mathrel{\widehat{=}} \textbf{any } t \textbf{ where } G \textbf{ during } C \textbf{ upon } F \textbf{ then } S \textbf{ end}$$

$$(\mathrm{cnc\_})\mathrm{f} \mathrel{\widehat{=}} \textbf{any } t \textbf{ where } H \textbf{ during } D \textbf{ upon } E \textbf{ then } R \textbf{ end}$$

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

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

$$(\Box C \wedge \Box\Diamond F) \;\Rightarrow\; \Diamond(\Box D \wedge \Box\Diamond E)$$

Schedules Weakening

Practical Rules

$$(\Box C \;\wedge\; \Box\Diamond F) \quad\Rightarrow\quad \Diamond(\Box D \;\wedge\; \Box\Diamond E) \qquad\qquad \text{(REF\_LIVE)}$$

Schedules Weakening

Practical Rules

$$(\Box C \ \wedge \ \Box\Diamond F) \quad \Rightarrow \quad \Diamond(\Box D \ \wedge \ \Box\Diamond E) \qquad \text{(REF\_LIVE)}$$

- Practical rules:
  - Coarse-schedule following

$$C \wedge F \ \leadsto \ D \qquad \text{(C\_FLW)}$$

  - Coarse-schedule stabilising

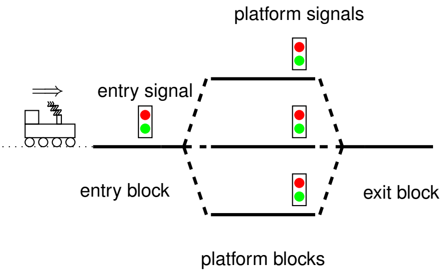
$$D \ \mathbf{un} \ \neg C \qquad \text{(C\_STB)}$$

  - Fine-schedule following

$$C \wedge F \ \leadsto \ E \qquad \text{(F\_FLW)}$$

A Signal Control System. The First Refinement

The State



- Introduce the network topology: *BLK*, *Entry*, *PLF*, *Exit*.
- Variable *loc* denotes location of trains in the network.

$$\text{inv1\_1} : loc \in trns \rightarrow BLK$$

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
$trns := trns \setminus \{t\}$
$loc := \{t\} \lhd loc$
end

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

$$t \in trns \;\rightsquigarrow\; t \in trns \land loc.t = Exit \qquad (\text{prg1\_1})$$

A Signal Control System. The First Refinement

New Event moveout

$$t \in trns \wedge loc.t \in PLF \;\leadsto\; t \in trns \wedge loc.t = Exit$$

$\Leftarrow$ Ensure rule

$$t \in trns \wedge loc.t \in PLF \;\mathbf{un}\; t \in trns \wedge loc.t = Exit \;\wedge$$
$$(\mathbf{tr}\,(t \in trns \wedge loc.t \in PLF) \wedge \neg(t \in trns \wedge loc.t = Exit))$$

$\Leftrightarrow$ Logic

$$\ldots \;\wedge\; (\mathbf{tr}\; t \in trns \wedge loc.t \in PLF)$$

A Signal Control System. The First Refinement

New Event moveout

$$t \in trns \wedge loc.t \in PLF \;\leadsto\; t \in trns \wedge loc.t = Exit$$

$\Leftarrow$ Ensure rule

$$t \in trns \wedge loc.t \in PLF \;\textbf{un}\; t \in trns \wedge loc.t = Exit \;\wedge$$
$$(\textbf{tr}\,(t \in trns \wedge loc.t \in PLF) \wedge \neg(t \in trns \wedge loc.t = Exit))$$

$\Leftrightarrow$ Logic

$$\ldots \;\wedge\; (\textbf{tr}\; t \in trns \wedge loc.t \in PLF)$$

moveout
any t **where**
$t \in trns \wedge loc.t \in PLF$
during
$t \in trns \wedge loc.t \in PLF$
then
$loc.t := Exit$
end

A Signal Control System. The Second Refinement

The State

SAF 1 There is at most one train on each block

invariants :

$$\forall t_1, t_2 \cdot t_1 \in trns \land t_2 \in trns \land loc.t_1 = loc.t_2 \Rightarrow t_1 = t_2$$

A Signal Control System. The Second Refinement

Refinement of moveout

(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

(cnc_)moveout
any t **where**
$t \in trns \land loc.t \in PLF \land$

during
$t \in trns \land loc.t \in PLF$
upon

then
$loc.t := Exit$
end

A Signal Control System. The Second Refinement

Refinement of moveout

(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

(cnc_)moveout
any t **where**
$t \in trns \land loc.t \in PLF \land$
$Exit \notin \operatorname{ran}.loc$
during
$t \in trns \land loc.t \in PLF$
upon

then
$loc.t := Exit$
end

A Signal Control System. The Second Refinement

Refinement of moveout

(cnc_)moveout
any t **where**
$t \in trns \wedge loc.t \in PLF \wedge$
$Exit \notin \mathrm{ran}\,.loc$
during
$t \in trns \wedge loc.t \in PLF$
upon
$Exit \notin \mathrm{ran}\,.loc$
then
$loc.t := Exit$
end

(abs_)moveout
any t **where**
$t \in trns \wedge loc.t \in PLF$
during
$t \in trns \wedge loc.t \in PLF$
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.

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