Systems Design Guided by Progress Concerns

2013-06-13

Systems Design Guided by Progress Concerns

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

Systems Design Guided by Progress Concerns

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

YORK UNIVERSITÉ UNIVERSITY

ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Safety vs. Liveness
Safety Properties
- Something (bad) never happens.
- e.g. invariance properties
Liveness Properties
- Something (good) will happen
- e.g. termination, progress

Safety vs. Liveness

Safety Properties

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

Liveness Properties

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

Safety vs. Liveness

Safety Properties

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

Liveness Properties

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

- Liveness properties are essential.

Safety vs. Liveness

Safety Properties

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

Liveness Properties

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

- Liveness properties are essential.

Safety vs. Liveness

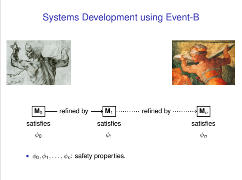
Safety Properties

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

Liveness Properties

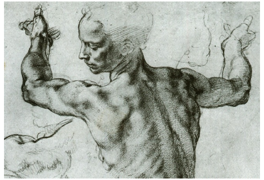
- Something (good)
  will happen
- e.g. termination, progress

- Liveness properties are essential.



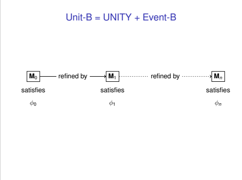
- To develop a system **M** satisfying property ϕ, i.e., $\mathbf{M} \models \phi$.
  - **M**: some transition system
  - ϕ: some logical formula
- The main challenge: the complexity of the system.
- Refinement allows the step-by-step design of the system.

Systems Development using Event-B



$$\begin{array}{ccccc} \boxed{\mathbf{M}_0} & \text{— refined by} \longrightarrow & \boxed{\mathbf{M}_1} & \cdots\cdots \text{ refined by } \cdots\cdots\rightarrow & \boxed{\mathbf{M}_n} \\ \text{satisfies} & & \text{satisfies} & & \text{satisfies} \\ \phi_0 & & \phi_1 & & \phi_n \end{array}$$

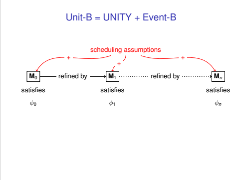
- $\phi_0, \phi_1, \ldots, \phi_n$: safety properties.



- 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.

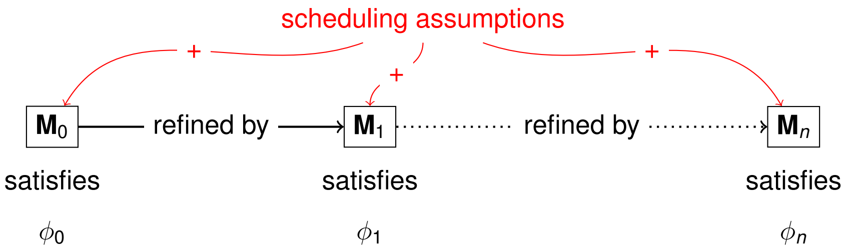
Unit-B = UNITY + Event-B

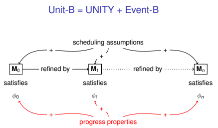
$$\begin{array}{ccccc} \boxed{\mathbf{M}_0} & \xrightarrow{\text{refined by}} & \boxed{\mathbf{M}_1} & \cdots\cdots \text{ refined by } \cdots\cdots\rightarrow & \boxed{\mathbf{M}_n} \\ \text{satisfies} & & \text{satisfies} & & \text{satisfies} \\ \phi_0 & & \phi_1 & & \phi_n \end{array}$$



- 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.

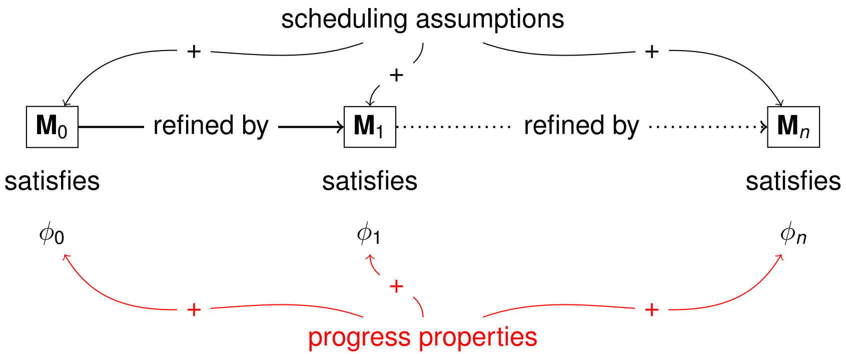
Unit-B = UNITY + Event-B

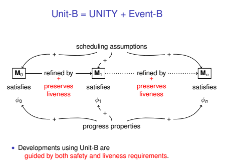




- Inspired by UNITY and Event-B.
- Support the reasoning of liveness properties (UNITY).
- Refinement of transition systems (Event-B style).
- Developments using Unit-B are
  guided by both safety and liveness requirements.

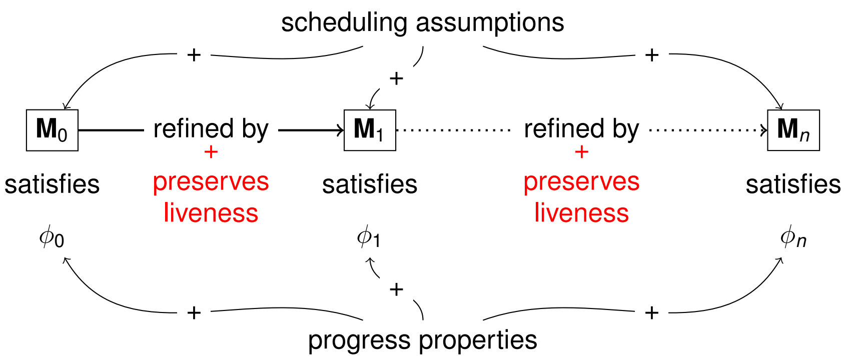
Unit-B = UNITY + Event-B





- 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.

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 formulae can be extended by combining the Boolean operators $(\neg, \wedge, \vee, \Rightarrow)$ with temporal operators:

Traces and the Language of Temporal Logic

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

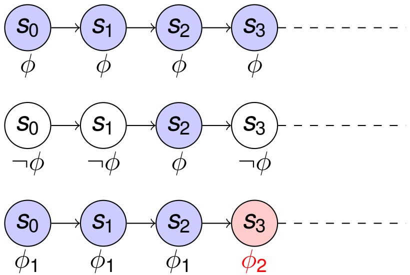
$$\sigma = s_0, s_1, s_2, s_3, \ldots$$

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

always: $\Box\,\phi$

eventually: $\Diamond\,\phi$

until: $\phi_1\,\mathcal{U}\,\phi_2$



Unit-B Models. Guarded and Scheduled Events

e
any t where
$G.t.v$

then
$S.t.v.v'$
end

- Execution of e.t corresponds to a formula $act.(e.t)$.

- 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 $act.(e.t)$.

Unit-B Models. Guarded and Scheduled Events

e
any t **where**
$G.t.v$

then
$S.t.v.v'$
end

- Execution of e.t corresponds to a formula $act.(e.t)$.

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*.(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 e.t is executed when $F.t.v$ holds.

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

- 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 *act*.(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*.(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 e.t is executed when $F.t.v$ holds.

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

- 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 $act.(\mathrm{e}.t)$.

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 when $F.t.v$ holds.

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

Schedules vs. Fairness

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

Schedules vs. Fairness
e $\widehat{=}$ any t where $G.t.v$ during $C.t.v$ upon $F.t.v$ then … 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 $\top$.

Conventions

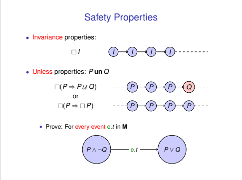
e $\widehat{=}$ **any** t **where** ... **during** $C.t.v$ **upon** $F.t.v$ **then** ... **end**

- 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 $\top$.

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 $\top$.



Invariance properties are proved using induction technique.

- I holds for every reachable state.
- Proved using the standard induction technique.

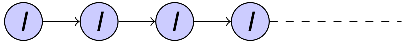
If P holds at some point then

- Q never holds and P continues to hold forever, or
- Q holds eventually and P continues to hold at least until Q holds.

Safety Properties

- Invariance properties:

$\Box I$

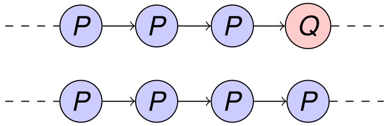


- Unless properties: P **un** Q

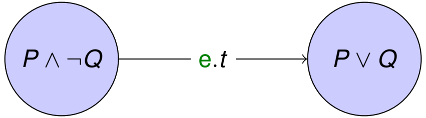
$\Box(P \Rightarrow P \,\mathcal{U}\, Q)$

or

$\Box(P \Rightarrow \Box P)$



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



Liveness Properties
- Progress properties
$$P \leadsto Q \;\widehat{=}\; \Box(P \Rightarrow \Diamond Q)$$
- Some important rules
$$(P \Rightarrow Q) \Rightarrow (P \leadsto Q) \quad \text{(Implication)}$$
$$(P \leadsto Q) \land (Q \leadsto R) \Rightarrow (P \leadsto R) \quad \text{(Transitivity)}$$

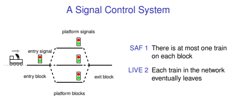
Liveness Properties

- Progress properties

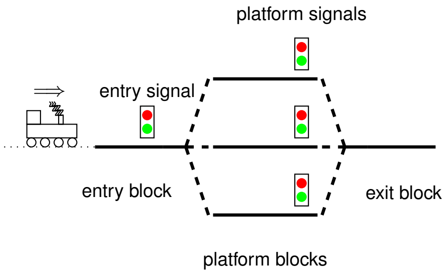
$$P \leadsto Q \;\widehat{=}\; \Box(P \Rightarrow \Diamond Q)$$

- Some important rules

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



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

platform signals
entry signal
entry block
exit block
platform blocks

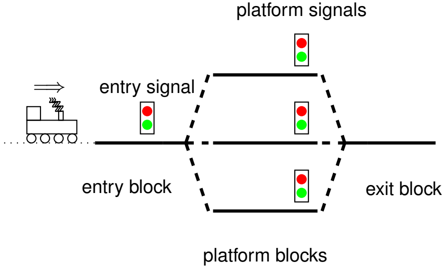
SAF 1 There is at most one train on each block

LIVE 2 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 specification for the controller

A Signal Control System

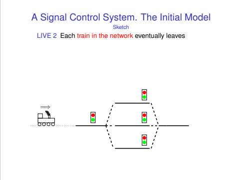


SAF 1 There is at most one train on each block

LIVE 2 Each train in the network eventually leaves

Refinement Strategy

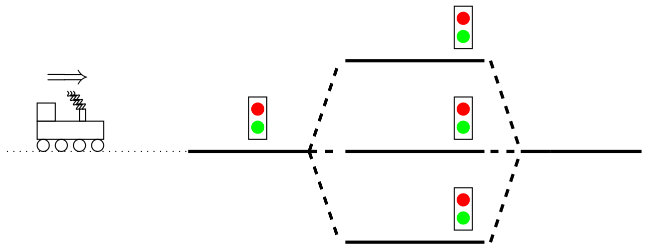
- Model 0 To model trains in the network, focus on LIVE 2
- Ref. 1 To introduce the network topology
- Ref. 2 To take into account SAF 1
- Ref. 3 To introduce signals and derive a specification for the controller

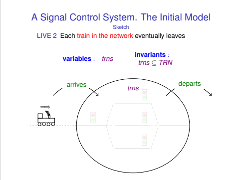


A Signal Control System. The Initial Model

Sketch

LIVE 2 Each train in the network eventually leaves





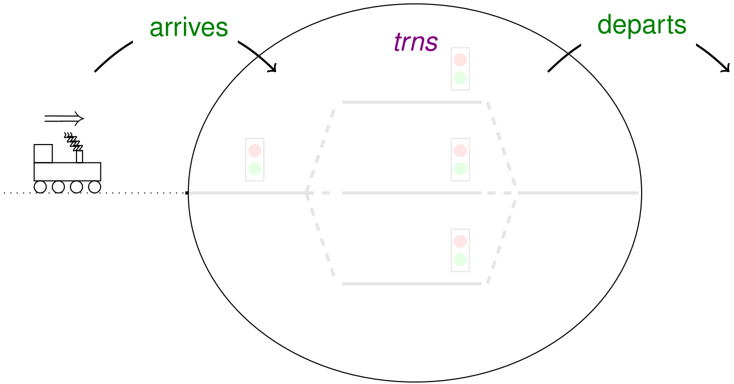
A Signal Control System. The Initial Model

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 :
$trns \subseteq TRN$

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

departs
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.

A Signal Control System. The Initial Model

Sketch

LIVE 2 Each train in the network eventually leaves

variables : $trns$

invariants : $trns \subseteq TRN$

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

departs 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**
e $\widehat{=}$ **any** t **where** $G.t.v$ **during** $C.t.v$ **upon** $F.t.v$ **then** $S.t.v.v'$ **end**
such that
$\Box(P \Rightarrow C)$, (SCH)
$C \leadsto F$, (PRG)
(NEG)
Note: general progress properties can be proved using the *induction* or *ensure* rules.

- (SCH) is trivial when C is P
- (PRG) is trivial when F is $\top$

Transient Properties

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

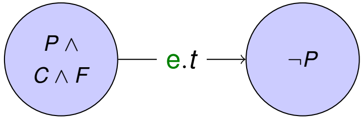
M *satisfies* $P \leadsto \neg P$ *if there exists an event in* **M**

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

such that

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

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



(NEG)

- Note: general progress properties can be proved using the *induction* or *ensure* rules.

A Signal Control System. The Initial Model
Properties
departs
any t where
$t \in TRN$
prg0_1 : $t \in trns \leadsto t \notin trns$
then
$trns := trns \setminus \{t\}$
end

prg0$_1$ holds no matter what the other events of the system are. In particular, we did not need to say that eventually, depart is the only event enabled. It is going to be executed because it is scheduled to. This is one strength of (weak) fairness.

A Signal Control System. The Initial Model

Properties

departs
  any t **where**
    $t \in TRN$

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

  then
    $trns := trns \setminus \{t\}$
  end

A Signal Control System. The Initial Model
Properties
departs 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)

$prg0_1$ holds no matter what the other events of the system are. In particular, we did not need to say that eventually, depart is the only event enabled. It is going to be executed because it is scheduled to. This is one strength of (weak) fairness.

A Signal Control System. The Initial Model

Properties

```
departs
  any t where
    t ∈ TRN
  during
    t ∈ trns
  then
    trns := trns \ {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

- Event-based reasoning.

(abs_)e $\widehat{=}$ **any** t **where** G **during** C **upon** F **then** S **end**
(cnc_)f $\widehat{=}$ **any** t **where** H **during** D **upon** E **then** R **end**

- Safety:
  - Guard strengthening: $H \Rightarrow G$
  - Action strengthening: $R \Rightarrow S$
- Liveness:
  - Scheduling assumptions strengthening.
  - Schedules weakening:

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

Refinement

- Event-based reasoning.

(abs_)e $\widehat{=}$ **any** t **where** G **during** C **upon** F **then** S **end**

(cnc_)f $\widehat{=}$ **any** t **where** H **during** D **upon** E **then** R **end**

- Safety:
  - Guard strengthening: $H \Rightarrow G$
  - Action strengthening: $R \Rightarrow S$
- Liveness:
  - Scheduling assumptions strengthening.
  - Schedules weakening:

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

Schedules Weakening
Practical Rules
$(\Box C \wedge \Box\Diamond F) \Rightarrow \Diamond(\Box D \wedge \Box\Diamond E)$ (REF_LIVE)

This rule

$$D \wedge E \Rightarrow F \qquad (1)$$

Schedules Weakening

Practical Rules

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

Schedules Weakening
Practical Rules
$$(\Box C \wedge \Box\Diamond F) \;\Rightarrow\; \Diamond(\Box D \wedge \Box\Diamond E) \qquad \text{(REF\_LIVE)}$$
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
$$C \wedge F \;\leadsto\; E \qquad \text{(F\_FLW)}$$

This rule

$$D \wedge E \;\Rightarrow\; F \tag{2}$$

Schedules Weakening

Practical Rules

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

Practical rules

- Coarse-schedule following

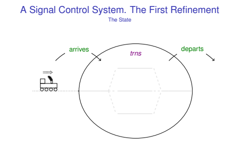
$$C \wedge F \;\leadsto\; D \qquad \text{(C\_FLW)}$$

- Coarse-schedule stabilising

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

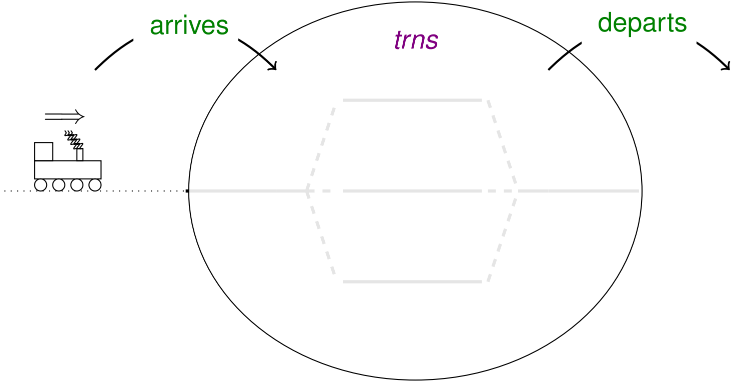
- Fine-schedule following

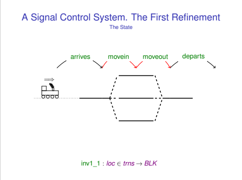
$$C \wedge F \;\leadsto\; E \qquad \text{(F\_FLW)}$$



A Signal Control System. The First Refinement

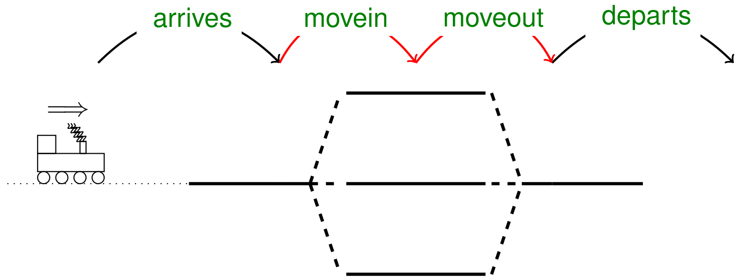
The State





A Signal Control System. The First Refinement

The State



inv1_1 : $loc \in trns \to BLK$

A Signal Control System. The First Refinement
Refinement of departs
(abs_)departs any t where $t \in TRN$ during $t \in trns$ then $trns := trns \setminus \{t\}$ end
(cnc_)departs 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):
$t \in trns \;\leadsto\; t \in trns \land loc.t = Exit$ (prg1_1)

A Signal Control System. The First Refinement

Refinement of departs

(abs_)departs
any t **where**
  $t \in TRN$
during
  $t \in trns$
then
  $trns := trns \setminus \{t\}$
end

(cnc_)departs
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):

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

A Signal Control System. The First Refinement

New Event moveout

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

A Signal Control System. The First Refinement

New Event moveout

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

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

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

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

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

  then
    $loc.t := Exit$
  end

A Signal Control System. The Second Refinement
Refinement of moveout

A Signal Control System. The Second Refinement

Refinement of moveout

(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

(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

then
$loc.t := Exit$
end

A Signal Control System. The Second Refinement
Refinement of moveout
(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
(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
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 moveout

(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

(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

- 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.

Future Work
- Decomposition / Composition
- Tool support

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.

Future Work

- Decomposition / Composition
- Tool support

Refinement

The UNITY way vs. the Event-B way

- UNITY: Refines the formulae.

$$\overbrace{\phi \Leftarrow \phi_1 \Leftarrow \ldots \Leftarrow \phi_n}^{\text{Refinement}} \underbrace{\dashv \quad \mathbf{M}}_{\text{Translation}}$$

- Event-B: Refines transition systems.

$$\underbrace{\phi \quad \dashv}_{\text{Verification}} \overbrace{\mathbf{M}_0 \text{ refined by } \mathbf{M}_1 \ldots \text{ refined by } \mathbf{M}}^{\text{Refinement}}$$

Refinement

The UNITY way vs. the Event-B way

- UNITY: Refines the formulae.

$$\overbrace{\phi \Leftarrow \phi_1 \Leftarrow \ldots \Leftarrow \phi_n}^{\text{Refinement}} \quad \underbrace{\phi_n \Dashv \mathbf{M}}_{\text{Translation}}$$

  - Cons: Hard to understand the choice of refinement.

- Event-B: Refines transition systems.

$$\underbrace{\phi \Dashv \mathbf{M}_0}_{\text{Verification}} \quad \overbrace{\mathbf{M}_0 \text{ refined by } \mathbf{M}_1 \ldots \text{ refined by } \mathbf{M}}^{\text{Refinement}}$$

  - Cons: No support for liveness properties.

Execution of Unit-B Models

$$\begin{aligned} ex.\mathbf{M} &= saf.\mathbf{M} \land live.\mathbf{M} && (3)\\ saf.\mathbf{M} &= init.\mathbf{M} \land \Box\, step.\mathbf{M} && (4)\\ step.\mathbf{M} &= (\exists \mathrm{e}.t \in \mathbf{M} \cdot act.(\mathrm{e}.t)) \lor \textsc{skip} && (5)\\ live.\mathbf{M} &= \forall \mathrm{e}.t \in \mathbf{M} \cdot sched.(\mathrm{e}.t) && (6)\\ sched.(\mathrm{e}.t) &= \Box(\Box\, C \land \Box\Diamond F \;\Rightarrow\; \Diamond(F \land act.(\mathrm{e}.t))) && (7) \end{aligned}$$

Execution of Unit-B Models

$$ex.\mathbf{M} = saf.\mathbf{M} \land live.\mathbf{M} \tag{3}$$

$$saf.\mathbf{M} = init.\mathbf{M} \land \Box\, step.\mathbf{M} \tag{4}$$

$$step.\mathbf{M} = (\exists \mathrm{e}.t \in \mathbf{M} \cdot act.(\mathrm{e}.t)) \lor \textsc{skip} \tag{5}$$

$$live.\mathbf{M} = \forall \mathrm{e}.t \in \mathbf{M} \cdot sched.(\mathrm{e}.t) \tag{6}$$

$$sched.(\mathrm{e}.t) = \Box(\Box\, C \land \Box\Diamond F \;\Rightarrow\; \Diamond(F \land act.(\mathrm{e}.t))) \tag{7}$$

The Ensure Rule

Theorem (The ensure-rule)
For all state predicates p and q,

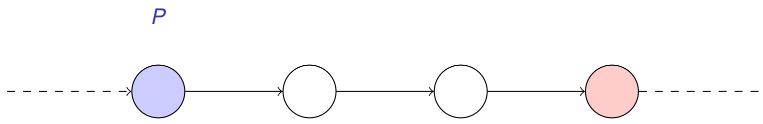
$$(P\ \mathbf{un}\ Q) \;\wedge\; ((P \wedge \neg Q) \leadsto (\neg P \vee Q)) \quad\Rightarrow\quad (P \leadsto Q) \qquad \text{(ENS)}$$

The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

$$(P\ \mathbf{un}\ Q) \;\wedge\; ((P \wedge \neg Q) \leadsto (\neg P \vee Q)) \quad\Rightarrow\quad (P \leadsto Q) \qquad \text{(ENS)}$$



The Ensure Rule

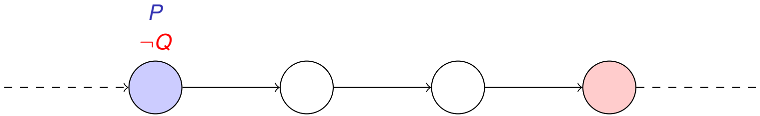
Theorem (The ensure-rule)
For all state predicates p and q,

$$(P\ \mathbf{un}\ Q) \ \wedge\ ((P \wedge \neg Q) \leadsto (\neg P \vee Q)) \quad \Rightarrow \quad (P \leadsto Q) \qquad \text{(ENS)}$$

The Ensure Rule

Theorem (The ensure-rule)
For all state predicates p and q,

$$(P\ \mathbf{un}\ Q) \ \wedge\ ((P \wedge \neg Q) \leadsto (\neg P \vee Q)) \quad \Rightarrow \quad (P \leadsto Q) \qquad \text{(ENS)}$$



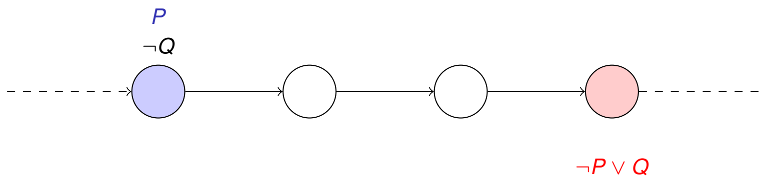
The Ensure Rule
Theorem (The ensure-rule)
For all state predicates p and q,
$(P \,\mathbf{un}\, Q) \wedge ((P \wedge \neg Q) \leadsto (\neg P \vee Q)) \Rightarrow (P \leadsto Q)$ (ENS)

The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

$$(P \,\mathbf{un}\, Q) \;\wedge\; ((P \wedge \neg Q) \leadsto (\neg P \vee Q)) \;\Rightarrow\; (P \leadsto Q) \qquad \text{(ENS)}$$



The Ensure Rule

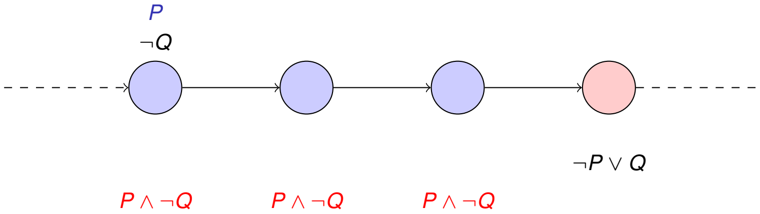
Theorem (The ensure-rule)
For all state predicates p and q,

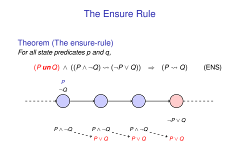
The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

$$(P\ \mathbf{un}\ Q) \ \wedge\ ((P \wedge \neg Q) \leadsto (\neg P \vee Q)) \quad \Rightarrow \quad (P \leadsto Q) \qquad \text{(ENS)}$$



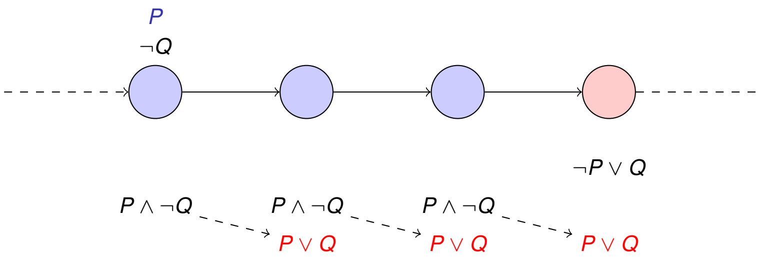


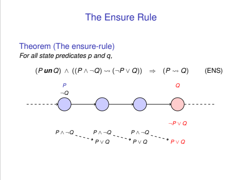
The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

$$(P\ \mathbf{un}\ Q) \ \land\ ((P \land \neg Q) \leadsto (\neg P \lor Q)) \ \Rightarrow\ (P \leadsto Q) \qquad \text{(ENS)}$$



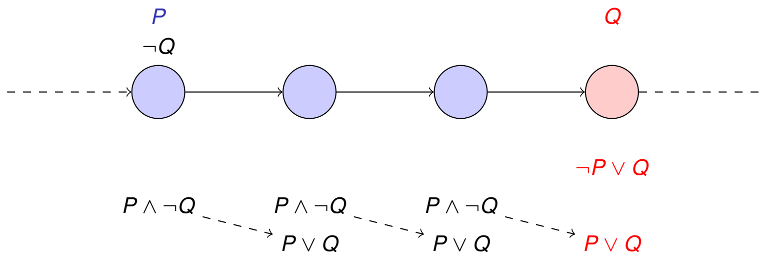


The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

$$(P \ \mathbf{un}\ Q) \ \wedge\ ((P \wedge \neg Q) \leadsto (\neg P \vee Q)) \quad\Rightarrow\quad (P \leadsto Q) \qquad \text{(ENS)}$$



The specification of the controller

ctrl_platform
any p **where**
$p \in PLF \land p \in \mathrm{ran}\,.loc \land Exit \notin \mathrm{ran}\,.loc \land$
$\forall q \cdot q \in PLF \Rightarrow sgn.q = RD$
during
$p \in PLF \land p \in \mathrm{ran}\,.loc \land sgn.p = RD$
upon
$Exit \notin \mathrm{ran}(loc) \land \forall q \cdot q \in PLF \land q \neq p \Rightarrow sgn.q = RD$
then
$sgn.p := GR$
end

The specification of the controller

ctrl_platform

any p **where**

$p \in PLF \land p \in \mathrm{ran}\,.loc \land Exit \notin \mathrm{ran}\,.loc \land$

$\forall q \cdot q \in PLF \Rightarrow sgn.q = RD$

during

$p \in PLF \land p \in \mathrm{ran}\,.loc \land sgn.p = RD$

upon

$Exit \notin \mathrm{ran}(loc) \land \forall q \cdot q \in PLF \land q \neq p \Rightarrow sgn.q = RD$

then

$sgn.p := GR$

end