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

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

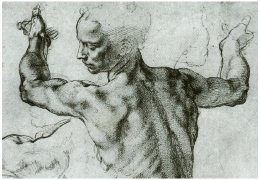
- Something (bad)
  never happens.
- e.g. invariance properties

Liveness Properties

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

- Liveness properties are essential.

Systems Development using Event-B



$$\boxed{\mathbf{M}_0} \xrightarrow{\text{refined by}} \boxed{\mathbf{M}_1} \cdots\cdots \text{refined by} \cdots\cdots \boxed{\mathbf{M}_n}$$

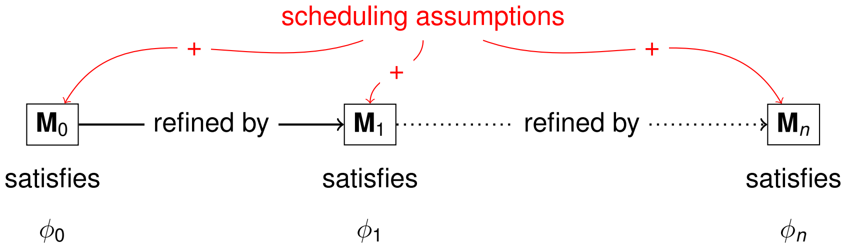
satisfies	satisfies	satisfies
ϕ_0	ϕ_1	ϕ_n

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

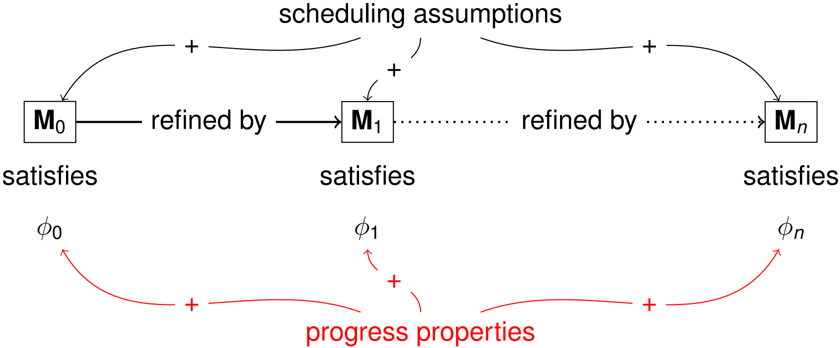
Unit-B = UNITY + Event-B

$$\begin{tikzcd}
\boxed{\mathbf{M}_0} \arrow[r, "\text{refined by}"] & \boxed{\mathbf{M}_1} \arrow[r, dotted, "\text{refined by}"] & \boxed{\mathbf{M}_n} \\
\text{satisfies} & \text{satisfies} & \text{satisfies} \\
\phi_0 & \phi_1 & \phi_n
\end{tikzcd}$$

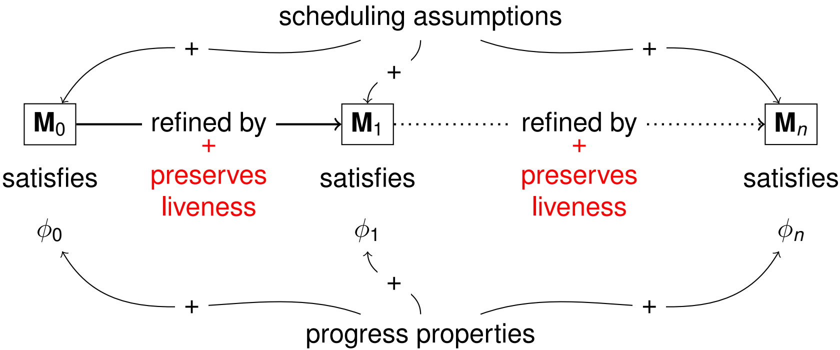
Unit-B = UNITY + Event-B



Unit-B = UNITY + Event-B



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

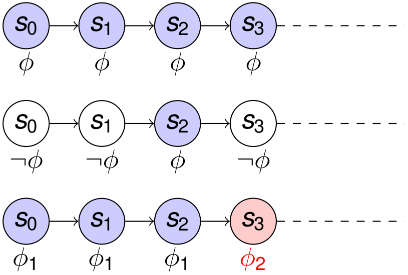
$$\sigma = s_0, s_1, s_2, s_3, \ldots$$

- 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:

always: $\square\, \phi$

eventually: $\lozenge\, \phi$

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 $act.(e.t)$.

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.

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.

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

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 $\top$.

Conventions

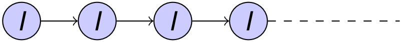
$e \mathrel{\widehat{=}}$ **any** t **where** $\ldots$ **during** $C.t.v$ **upon** $F.t.v$ **then** $\ldots$ **end**

- 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 $\top$.

Safety Properties

- Invariance properties:

$$\Box I$$

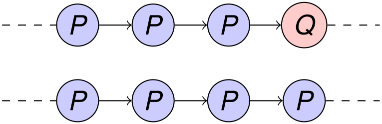


- Unless properties: $P \ \mathbf{un} \ Q$

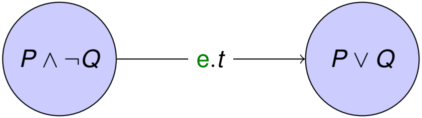
$$\Box(P \Rightarrow P \,\mathcal{U}\, Q)$$

or

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



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



Liveness Properties

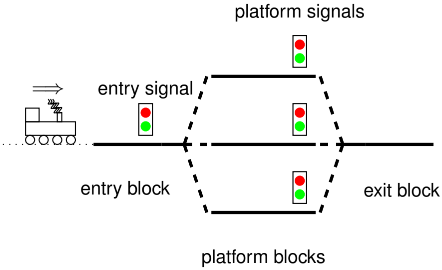
- Progress properties

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

- Some important rules

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

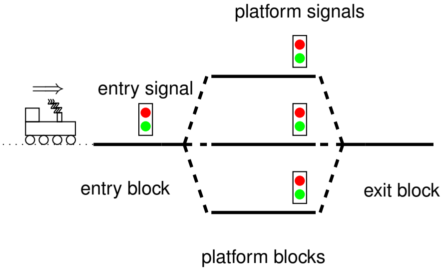
A Signal Control System



SAF 1 There is at most one train on each block

LIVE 2 Each train in the network eventually leaves

A Signal Control System



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

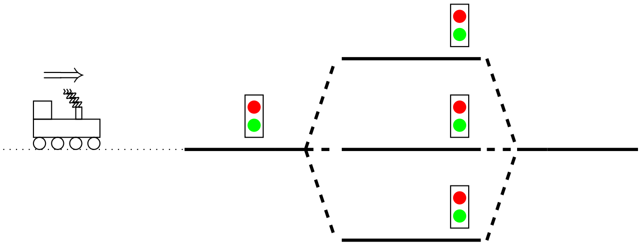
Ref. 2 To take into account SAF 1

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



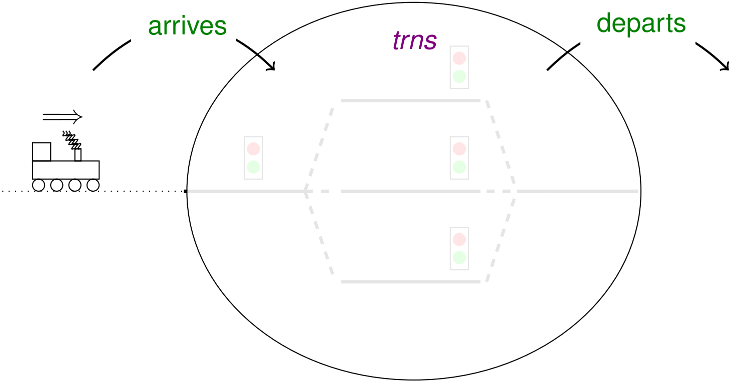
A Signal Control System. The Initial Model

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.

Transient Properties

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

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

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

such that

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

$$C \leadsto F\ , \tag{PRG}$$

$$P \wedge C \wedge F \xrightarrow{\mathsf{e}.t} \neg P \tag{NEG}$$

- 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

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)

Refinement

- Event-based reasoning.

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

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

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

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

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

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

- Coarse-schedule stabilising

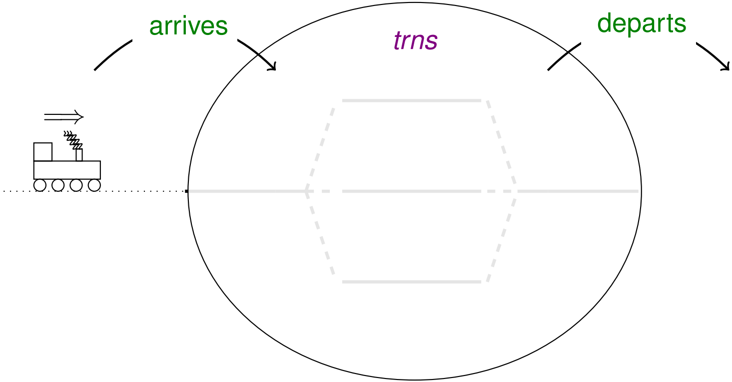
$$D \ \mathbf{un} \ \neg C \qquad \text{(C\_STB)}$$

- Fine-schedule following

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

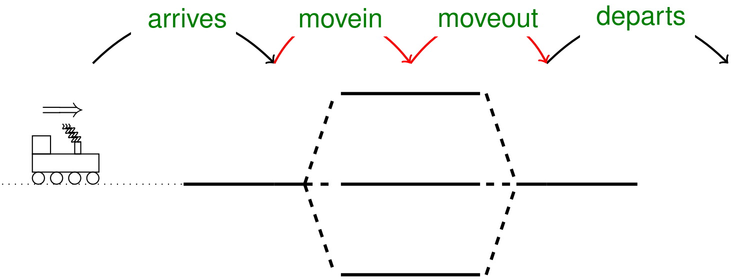
A Signal Control System. The First Refinement

The State



A Signal Control System. The First Refinement

The State



inv1_1 : $loc \in trns \rightarrow BLK$

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):

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

A Signal Control System. The First Refinement

New Event moveout

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

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

(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.

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.

$$\overbrace{\phi \Leftarrow \phi_1 \Leftarrow \ldots \Leftarrow \phi_n}^{\text{Refinement}} \hspace{-2em} \underbrace{ \dashv \mathbf{M}}_{\text{Translation}}$$

- Event-B: Refines transition systems.

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

Refinement

The UNITY way vs. the Event-B way

- UNITY: Refines the formulae.

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

  - Cons: Hard to understand the choice of refinement.

- Event-B: Refines transition systems.

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

  - Cons: No support for liveness properties.

Execution of Unit-B Models

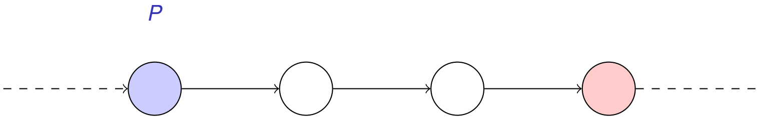
$$
\begin{aligned}
ex.\mathbf{M} &= saf.\mathbf{M} \land live.\mathbf{M} && (1)\\
saf.\mathbf{M} &= init.\mathbf{M} \land \Box\, step.\mathbf{M} && (2)\\
step.\mathbf{M} &= (\exists \mathrm{e}.t \in \mathbf{M} \cdot act.(\mathrm{e}.t)) \lor \textsc{skip} && (3)\\
live.\mathbf{M} &= \forall \mathrm{e}.t \in \mathbf{M} \cdot sched.(\mathrm{e}.t) && (4)\\
sched.(\mathrm{e}.t) &= \Box(\Box\, C \land \Box \Diamond F \Rightarrow \Diamond(F \land act.(\mathrm{e}.t))) && (5)
\end{aligned}
$$

The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

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

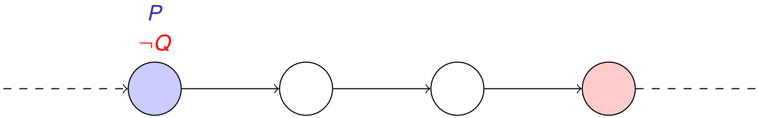


The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

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

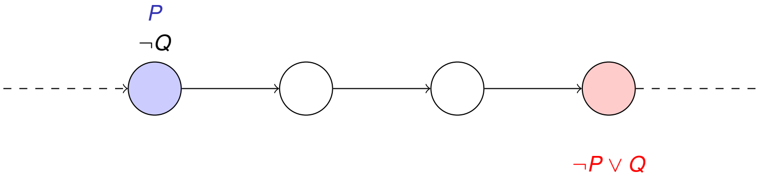


The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

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

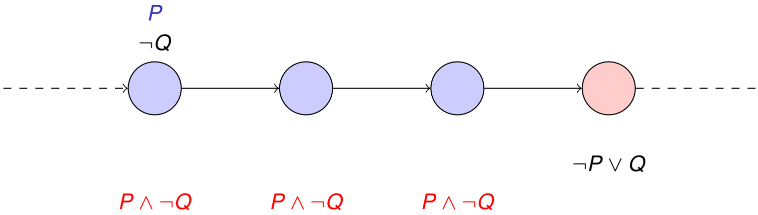


The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

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

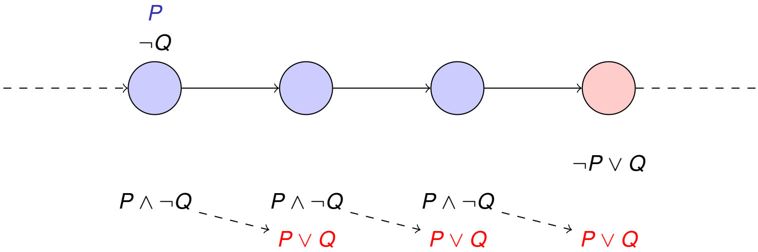


The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

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

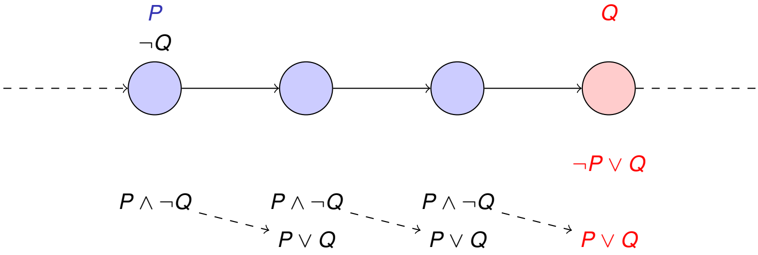


The Ensure Rule

Theorem (The ensure-rule)

For all state predicates p and q,

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



The specification of the controller

ctrl_platform

any p **where**

$p \in PLF \wedge p \in \operatorname{ran}.loc \wedge Exit \notin \operatorname{ran}.loc \wedge$

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

during

$p \in PLF \wedge p \in \operatorname{ran}.loc \wedge sgn.p = RD$

upon

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

then

$sgn.p := GR$

end