Formal Developments of Examples in the Paper:"Formalizing Hybrid Systems with Event-B and the Rodin Platform"

Author: Wen Su, Jean-Raymond Abrial, and Huibiao Zhu

Oct 2013

Contents

Chapter 1

The Saw

1.1 Machine

1.1.1 m0

An Event-B Specification of m0
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE m0
VARIABLES

x
up

INVARIANTS

inv0_1 : $x \in \{0, 1\}$
inv0_2 : $up \in BOOL$

EVENTS
Initialisation
begin

act1 : $x := 0$
act2 : $up := FALSE$

end
Event $UP \widehat{=}$
when

grd1 : $x = 0$
grd2 : $up = FALSE$

then

act1 : $x := 1$
act2 : $up := TRUE$

end
Event $DN \widehat{=}$
when

grd1 : $x = 1$

grd2 : $up = TRUE$

then

act1 : $x := 0$

act2 : $up := FALSE$

end

END

1.1.2 m1

An Event-B Specification of m1
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE m1

REFINES m0

VARIABLES

x_c

now

b

up

INVARIANTS

inv1_1 : $x\_c \in \mathbb{N} \pfun \mathbb{Z}$

inv1_2 : $now \in dom(x\_c)$

inv1_3 : $b = FALSE \Rightarrow x\_c(now) = x$

inv1_11 : $b = TRUE \Rightarrow now + 1 \in dom(x\_c)$

inv1_4 : $b = TRUE \Rightarrow x\_c(now + 1) = x$

inv1_5 : $b = TRUE \land up = TRUE \Rightarrow (x\_c = \lambda t \cdot t \in \mathbb{N} \land t \geq now \land t \leq now + 1 | t - now)$

inv1_6 : $b = TRUE \land up = FALSE \Rightarrow (x\_c = \lambda t \cdot t \in \mathbb{N} \land t \geq now \land t \leq now + 1 | 1 - t + now)$

inv1_7 : $b = FALSE \land up = TRUE \Rightarrow (x\_c = \lambda t \cdot t \in \mathbb{N} \land t \geq now - 1 \land t \leq now | t - now + 1)$

inv1_8 : $b = FALSE \land up = FALSE \Rightarrow (x\_c = \lambda t \cdot t \in \mathbb{N} \land t \geq now - 1 \land t \leq now | -t + now)$

thm1_1 : $b = TRUE \Rightarrow min(\{t | t \in dom(x\_c) \land t \geq now \land ((x\_c(t) = 0 \land up = FALSE) \lor (x\_c(t) = 1 \land up = TRUE))\}) = now + 1$

inv1_9 : $ran(x\_c) \subseteq \{0, 1\}$

inv1_10 : $b = TRUE \Rightarrow \neg((x\_c(now) = 0 \land up = FALSE) \lor (x\_c(now) = 1 \land up = TRUE))$

EVENTS

Initialisation

begin

act3 : $x\_c := \{0 \mapsto 0\}$

act2 : $now := 0$

act5 : $up := FALSE$

act4 : $b := FALSE$

end

Event $UP \widehat{=}$
refines UP
when

grd1 : $x\_c(now) = 0$
grd3 : $up = FALSE$
grd2 : $b = FALSE$

then

act1 : $x\_c := \lambda t \cdot t \in \mathbb{N} \land t \geq now \land t \leq now + 1 | t - now$
act3 : $up := TRUE$
act2 : $b := TRUE$

end
Event $DN \widehat{=}$
refines DN
when

grd1 : $x\_c(now) = 1$
grd3 : $up = TRUE$
grd2 : $b = FALSE$

then

act1 : $x\_c := \lambda t \cdot t \in \mathbb{N} \land t \geq now \land t \leq now + 1 | 1 - t + now$
act3 : $up := FALSE$
act2 : $b := TRUE$

end
Event $click \widehat{=}$
when

grd1 : $b = TRUE$

then

act1 : $b := FALSE$
act2 : $now := now + 1$

end
END

1.1.3 m1_alternative

An Event-B Specification of m1_alternative
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE m1_alternative
REFINES m0
VARIABLES

x_c
now
up

INVARIANTS

inv2_1 : $x\_c \in \mathbb{N} \pfun \mathbb{Z}$

inv2_0 : $now \in \mathbb{N}$

inv2_2 : $now \in dom(x\_c)$

inv2_3 : $x = x\_c(now)$

inv2_4 : $up = TRUE \Rightarrow (x\_c = \lambda t \cdot t \in \mathbb{N} \land t \geq now - 1 \land t \leq now | t - now + 1)$

inv2_5 : $up = FALSE \Rightarrow (x\_c = \lambda t \cdot t \in \mathbb{N} \land t \geq now - 1 \land t \leq now | - t + now)$

thm2_1 : $min(\{t | t \geq now \land ((1 - t + now = 0 \land up = FALSE) \lor (t - now = 1 \land up = TRUE))\}) = now + 1$

EVENTS

Initialisation

begin

act1 : $now := 0$

act2 : $x\_c := \{0 \mapsto 0\}$

act3 : $up := FALSE$

end

Event $UP \widehat{=}$

refines UP

when

grd1 : $x\_c(now) = 0$

grd2 : $up = FALSE$

then

act1 : $x\_c := \lambda t \cdot t \in \mathbb{N} \land t \geq now \land t \leq now + 1 | t - now$

act3 : $up := TRUE$

act2 : $now := now + 1$

end

Event $DN \widehat{=}$

refines DN

when

grd1 : $x\_c(now) = 1$

grd2 : $up = TRUE$

then

act1 : $x\_c := \lambda t \cdot t \in \mathbb{N} \land t \geq now \land t \leq now + 1 | 1 - t + now$

act3 : $up := FALSE$

act2 : $now := now + 1$

end

END

Chapter 2

Nuclear Plant Cooling

2.1 Context

2.1.1 C_c0

An Event-B Specification of C_c0
Creation Date: 1Nov2013 @ 04:45:41 PM

CONTEXT C_c0
CONSTANTS

$theta\_m$
$theta\_M$
$v1$
$v2$
vr
T
a
$b1$
$b2$

AXIOMS

axm0_1 : $theta\_m \in \mathbb{N}$
axm0_2 : $theta\_M \in \mathbb{N}$
axm0_3 : $theta\_M > theta\_m$
axm0_4 : $v1 > 0$
axm0_5 : $v2 > 0$
axm0_6 : $vr > 0$
axm0_7 : $T > 0$
axm1 : $a > 0$
axm2 : $b1 > 0$
axm3 : $b2 > 0$
axm0_8 : $vr * a = theta\_M - theta\_m$
axm0_9 : $v1 * b1 = theta\_M - theta\_m$

axm0_10 : $v2 * b2 = theta\_M - theta\_m$
axm0_11 : $a < T$
axm0_12 : $2 * a + b1 \geq T$
axm0_13 : $2 * a + b2 \geq T$

END

2.2 Machine

2.2.1 C_m0

An Event-B Specification of C_m0
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE C_m0
SEES C_c0
VARIABLES

$theta$
$t1$
$t2$
rod

INVARIANTS

inv0_1 : $theta \in \mathbb{N}$
inv0_2 : $t1 \in \mathbb{N}$
inv0_3 : $t2 \in \mathbb{N}$
inv0_4 : $rod \in \{0, 1, 2\}$
inv0_5 : $theta = theta\_M \Rightarrow t1 \geq T \vee t2 \geq T$
inv0_6 : $rod = 1 \Rightarrow t2 = a + b1$
inv0_7 : $rod = 2 \Rightarrow t1 = a + b2$
inv0_8 : $rod = 0 \Rightarrow t1 + a \geq T \vee t2 + a \geq T$
inv6 : $rod = 1 \Rightarrow t2 + a \geq T$ Needed to prove inv10
inv7 : $rod = 2 \Rightarrow t1 + a \geq T$ Needed to prove inv10
inv8 : $rod = 0 \wedge t1 \geq T \Rightarrow t2 = a$ Needed to prove inv8
inv9 : $rod = 0 \wedge t2 \geq T \Rightarrow t1 = a$ Needed to prove inv9
inv15 : $theta = theta\_M \Rightarrow rod = 0$

EVENTS
Initialisation
begin

act1 : $t1 := 2 * a + b2$
act2 : $t2 := a$
act4 : $theta := theta\_M$
act5 : $rod := 0$

end
Event *cool_rod1* $\widehat{=}$

when

grd2 : $t1 \geq T$
grd3 : $theta = theta\_M$

then

act1 : $rod := 1$
act3 : $t2 := t2 + b1$
act4 : $theta := theta\_m$

end

Event *release_rod1* $\widehat{=}$

when

grd1 : $rod = 1$
grd2 : $theta = theta\_m$

then

act1 : $rod := 0$
act2 : $t1 := a$
act3 : $t2 := t2 + a$
act4 : $theta := theta\_M$

end

Event *cool_rod2* $\widehat{=}$

when

grd2 : $t2 \geq T$
grd3 : $theta = theta\_M$

then

act1 : $rod := 2$
act3 : $t1 := t1 + b2$
act4 : $theta := theta\_m$

end

Event *release_rod2* $\widehat{=}$

when

grd1 : $rod = 2$
grd2 : $theta = theta\_m$

then

act1 : $rod := 0$
act2 : $t1 := t1 + a$
act3 : $t2 := a$
act4 : $theta := theta\_M$

end

END

2.2.2 C_m1

An Event-B Specification of C_m1
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE C_m1

REFINES C_m0

SEES C_c0

VARIABLES

$t1\_c$

$t2\_c$

$theta\_c$

now

rod

INVARIANTS

inv1_1 : $t1\_c \in \mathbb{N} \pfun \mathbb{Z}$

inv1_2 : $t2\_c \in \mathbb{N} \pfun \mathbb{Z}$

inv1_3 : $theta\_c \in \mathbb{N} \pfun \mathbb{Z}$

inv1_4 : $now \in dom(t1\_c) \cap dom(t2\_c) \cap dom(theta\_c)$

inv1_5 : $t1 = t1\_c(now)$

inv1_6 : $t2 = t2\_c(now)$

inv1_7 : $theta = theta\_c(now)$

thm1_1 : $rod = 1 \Rightarrow min(\{t | t \geq now \land t = now + b1\}) = now + b1$

thm1_2 : $rod = 2 \Rightarrow min(\{t | t \geq now \land t = now + b2\}) = now + b2$

thm1_3 : $rod = 0 \Rightarrow t1\_c(now) + a \geq T \lor t2\_c(now) + a \geq T$

thm1_4 : $rod = 0 \Rightarrow min(\{t | t \geq now \land (t1\_c(now) + t - now \geq T \lor t2\_c(now) + t - now \geq T) \land (t = now + a)\}) = now + a$

EVENTS

Initialisation

begin

act1 : $t1\_c := \{0 \mapsto 2 * a + b2\}$

act2 : $t2\_c := \{0 \mapsto a\}$

act3 : $rod := 0$

act5 : $now := 0$

act6 : $theta\_c := \{0 \mapsto theta\_M\}$

end

Event *cool_rod1* $\widehat{=}$

refines *cool_rod1*

when

grd2 : $t1\_c(now) \geq T$

grd3 : $theta\_c(now) = theta\_M$

grd4 : $b1 \in \mathbb{N}$

then

act1 : $rod := 1$

act3 : $t2\_c := \lambda t \cdot t \in now \mathinner{..} now + b1 | t2\_c(now) + (t - now)$

act5 : $t1\_c := \lambda t \cdot t \in now \mathinner{..} now + b1 | t1\_c(now)$

act6 : $theta\_c := \lambda t \cdot t \in now \mathinner{..} now + b1 | theta\_M - v1 * (t - now)$

act4 : $now := now + b1$

end

Event *release_rod1* $\widehat{=}$

refines *release_rod1*

when

grd1 : $rod = 1$
grd2 : $theta\_c(now) = theta\_m$
grd3 : $a \in \mathbb{N}$

then

act1 : $rod := 0$
act2 : $t1\_c := \lambda t \cdot t \in now \mathinner{..} now + a | (t - now)$
act3 : $t2\_c := \lambda t \cdot t \in now \mathinner{..} now + a | t2\_c(now) + (t - now)$
act5 : $theta\_c := \lambda t \cdot t \in now \mathinner{..} now + a | theta\_m + vr * (t - now)$
act4 : $now := now + a$

end

Event *cool_rod2* $\widehat{=}$

refines *cool_rod2*

when

grd2 : $t2\_c(now) \geq T$
grd3 : $theta\_c(now) = theta\_M$
grd4 : $b2 \in \mathbb{N}$

then

act1 : $rod := 2$
act3 : $t1\_c := \lambda t \cdot t \in now \mathinner{..} now + b2 | t1\_c(now) + (t - now)$
act5 : $t2\_c := \lambda t \cdot t \in now \mathinner{..} now + b2 | t2\_c(now)$
act6 : $theta\_c := \lambda t \cdot t \in now \mathinner{..} now + b2 | theta\_M - v2 * (t - now)$
act4 : $now := now + b2$

end

Event *release_rod2* $\widehat{=}$

refines *release_rod2*

when

grd1 : $rod = 2$
grd2 : $theta\_c(now) = theta\_m$
grd3 : $a \in \mathbb{N}$

then

act1 : $rod := 0$
act2 : $t1\_c := \lambda t \cdot t \in now \mathinner{..} now + a | t1\_c(now) + (t - now)$
act3 : $t2\_c := \lambda t \cdot t \in now \mathinner{..} now + a | (t - now)$
act5 : $theta\_c := \lambda t \cdot t \in now \mathinner{..} now + a | theta\_m + vr * (t - now)$
act4 : $now := now + a$

end

END

Chapter 3

Controlling Train

3.1 One Train Model

3.1.1 c0

An Event-B Specification of T_c0
Creation Date: 1Nov2013 @ 04:45:41 PM

CONTEXT T_c0

CONSTANTS

A
b
sl
e
m

AXIOMS

axm0_1 : $A > 0$
axm0_2 : $b > 0$
axm0_3 : $sl > 0$
axm0_4 : $e > 0$
axm0_5 : $m > 0$
axm5 : $\forall x, y \cdot \neg y = 0 \Rightarrow y * (x/y) = x$
axm6 : $b = 1$
axm7 : $A = 2$
axm8 : $A * b = 2$
axm9 : $e = 2$
axm11 : $\forall x \cdot x \in \mathbb{N} \Rightarrow 2 * (x * x/2) \in \mathbb{N}$
axm12 : $\forall x \cdot x \in \mathbb{N} \Rightarrow x * x/2 \in \mathbb{N}$

END

3.1.2 m0

An Event-B Specification of T_m0
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE T_m0
One train behind goal
SEES T_c0
VARIABLES

z

v

a

$phase$

INVARIANTS

inv0_1 : $z \in \mathbb{N}$
inv0_2 : $v \in \mathbb{N}$
inv0_3 : $a \in \{0, A, -b\}$
inv0_4 : $phase \in \{1, 2\}$
inv0_5 : $2 * b * (m - z) \geq v * v$
inv0_6 : $v \in 0 \,..\, sl$
inv6 : $m - z \geq 0$ SAFETY CONDITION: the train position z is always situated BEFORE the limit position m
inv8 : $phase = 2 \wedge v + a * e \leq 0 \Rightarrow a = -b$
inv9 : $phase = 2 \wedge a = A \Rightarrow 2 * b * (m - z) \geq v * v + (A * e * e + 2 * v * e) * (A + b)$
inv11 : $phase = 2 \wedge a = 0 \Rightarrow 2 * b * (m - z) \geq v * v + (A * e * e + 2 * v * e) * (A + b)$

EVENTS
Initialisation
begin

act1 : $phase := 1$
act3 : $z := 0$
act4 : $v := 0$
act5 : $a := 2$

end

Event *decide_1* $\widehat{=}$
Decelerate with -b
when

grd1 : $phase = 1$
grd2 : $2 * b * (m - z) < v * v + (A * e^2 + 2 * v * e) * (A + b)$
grd4 : $v > 0$

then

act1 : $a := -b$
act2 : $phase := 2$

end

Event *decide_2* $\widehat{=}$
Accelerate with A

when

grd1 : $phase = 1$
grd2 : $2 * b * (m - z) \geq v * v + (A * e * e + 2 * v * e) * (A + b)$
grd3 : $v + A * e \leq sl$
Resulting speed smaller than or equal to sl

then

act1 : $a := A$
a := A (A=2)
act2 : $phase := 2$

end

Event *decide_3* $\widehat{=}$
Acceleration 0

when

grd1 : $phase = 1$
grd2 : $2 * b * (m - z) \geq v * v + (A * e * e + 2 * v * e) * (A + b)$
grd3 : $v + A * e > sl$
Resulting speed greater than sl
grd7 : $v > 0$

then

act1 : $a := 0$
act2 : $phase := 2$

end

Event *drive_1* $\widehat{=}$
Reaching v=0

when

grd1 : $phase = 2$
grd2 : $v + a * e \leq 0$
grd3 : $a = -b$
grd4 : $2 * (v * v/2) = v * v$

then

act1 : $v := 0$
act2 : $z := z + (v * v)/2$
act3 : $phase := 1$

end

Event *drive_2* $\widehat{=}$
Proofs have to be done by cases (a=A, a=-b, or a=0)

when

grd2 : $phase = 2$
grd3 : $v + a * e > 0$
grd4 : $v \geq 0$

then

act1 : $v := v + a * e$
act2 : $z := z + v * e + a * e * e/2$
act3 : $phase := 1$

end

END

3.2 Multiple Train Model

3.2.1 c0

An Event-B Specification of c0
Creation Date: 1Nov2013 @ 04:45:41 PM

CONTEXT c0
SETS

ID

CONSTANTS

$trains$
A
b
e
sl

AXIOMS

axm1 : $trains \subseteq \mathbb{N}$
axm2 : $A > 0$
axm3 : $b > 0$
axm6 : $e > 0$
axm8 : $sl > 0$
axm4 : $b = 1$
axm5 : $A = 2$
axm7 : $e = 2$
axm9 : $\forall x, y \cdot \neg y = 0 \Rightarrow y * (x/y) = x$
axm10 : $\forall x \cdot x \in \mathbb{N} \Rightarrow 2 * (x * x/2) \in \mathbb{N}$
axm11 : $\forall x \cdot x \in \mathbb{N} \Rightarrow x * x/2 \in \mathbb{N}$
axm12 : $\forall x, y \cdot x \in \mathbb{Z} \wedge y \in \mathbb{Z} \wedge 2 * x \geq y * y \Rightarrow x \geq 0$

END

3.2.2 c1

An Event-B Specification of c1
Creation Date: 1Nov2013 @ 04:45:41 PM

CONTEXT c1
EXTENDS c0
CONSTANTS

c

AXIOMS

axm2 : $c > 0$
axm3 : $trains = 0 \mathinner{..} c$

END

3.2.3 m0

An Event-B Specification of m0
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE m0
SEES c0
VARIABLES

$phase$
$position$
acc
$speed$
$limit$

INVARIANTS

inv3 : $position \in trains \rightarrow \mathbb{Z}$
inv5 : $speed \in trains \rightarrow \mathbb{N}$
inv4 : $acc \in trains \rightarrow \{0, A, -b\}$
inv2 : $phase \in trains \rightarrow \{0, 1, 2\}$
inv6 : $limit \in trains \rightarrow \mathbb{N}$
inv7 : $\forall i \cdot i \in trains \Rightarrow 2 * b * (limit(i) - position(i)) \geq speed(i) * speed(i)$
inv12 : $\forall i \cdot i \in trains \Rightarrow speed(i) \in 0 \mathinner{..} sl$
inv8 : $\forall i \cdot i \in trains \Rightarrow limit(i) - position(i) \geq 0$
inv9 : $\forall i \cdot i \in trains \Rightarrow (phase(i) = 2 \wedge speed(i) + 2 * acc(i) \leq 0 \Rightarrow acc(i) = -b)$
inv10 : $\forall i \cdot i \in trains \Rightarrow (phase(i) = 2 \wedge acc(i) = A \Rightarrow 2 * (limit(i) - position(i))$
$\geq speed(i) * speed(i) + 24 + 12 * speed(i))$
inv11 : $\forall i \cdot i \in trains \Rightarrow (phase(i) = 2 \wedge acc(i) = 0 \Rightarrow 2 * limit(i) - 2 * position(i)$
$\geq speed(i) * speed(i) + 24 + 12 * speed(i))$
inv13 : $\forall i \cdot i \in trains \Rightarrow (phase(i) = 2 \wedge acc(i) = A \Rightarrow speed(i) + 4 \leq sl)$
inv14 : $\forall i \cdot i \in trains \Rightarrow (phase(i) = 2 \wedge speed(i) + e * acc(i) \leq 0 \Rightarrow acc(i) = -b)$
inv15 : $\forall i \cdot i \in trains \Rightarrow (phase(i) = 2 \wedge acc(i) = A \Rightarrow 2 * (limit(i) - position(i))$
$\geq speed(i) * speed(i) + (A * e * e + 2 * e * speed(i)) * (A + b))$
inv16 : $\forall i \cdot i \in trains \Rightarrow (phase(i) = 2 \wedge acc(i) = 0 \Rightarrow 2 * limit(i) - 2 * position(i)$
$\geq speed(i) * speed(i) + (A * e * e + 2 * e * speed(i)) * (A + b))$

EVENTS
Initialisation
begin

act2 : $phase := trains \times \{1\}$
act3 : $position := trains \times \{0\}$
act4 : $acc := trains \times \{0\}$
act5 : $speed := trains \times \{0\}$
act6 : $limit := trains \times \{0\}$

end
Event *get_limit* $\widehat{=}$

any

x

i

where

grd1 : $x \in 1 \mathinner{..} 300$
grd2 : $i \in trains$

then

act1 : $limit(i) := limit(i) + x$

end

Event *decide_1* $\widehat{=}$

any

i

where

grd5 : $i \in trains$
grd1 : $phase(i) = 1$
grd2 : $2 * limit(i) - 2 * position(i) < speed(i) * speed(i) + (A * e * e + 2 * e * speed(i)) * (A + b)$
grd3 : $A > 0$
grd4 : $speed(i) > 0$

then

act1 : $acc(i) := -b$
act2 : $phase(i) := 2$

end

Event *decide_2* $\widehat{=}$

any

i

where

grd7 : $i \in trains$
grd1 : $phase(i) = 1$
grd2 : $2 * limit(i) - 2 * position(i) \geq speed(i) * speed(i) + (A * e * e + 2 * e * speed(i)) * (A + b)$
grd3 : $speed(i) \geq 0$
grd4 : $e > 0$
grd5 : $A > 0$
grd6 : $speed(i) + e * A \leq sl$

then

act1 : $acc(i) := A$
act2 : $phase(i) := 2$

end

Event *decide_3* $\widehat{=}$

any

i

where

grd1 : $i \in trains$
grd2 : $phase(i) = 1$
grd3 : $2 * limit(i) - 2 * position(i) \geq speed(i) * speed(i) + (A * e * e + 2 * e * speed(i)) * (A + b)$

grd4 : $speed(i) > 0$
grd5 : $speed(i) + e * A > sl$

then

act1 : $acc(i) := 0$
act2 : $phase(i) := 2$

end

Event *drive1* $\widehat{=}$

any

i

where

grd1 : $i \in trains$
grd2 : $phase(i) = 2$
grd3 : $speed(i) + e * acc(i) \leq 0$
e=2

then

act1 : $speed(i) := 0$
act2 : $position(i) := position(i) + (speed(i) * speed(i))/(2 * b)$
act3 : $phase(i) := 1$

end

Event *drive2* $\widehat{=}$

any

i

where

grd1 : $i \in trains$
grd2 : $phase(i) = 2$
grd3 : $speed(i) + e * acc(i) > 0$
e=2
grd4 : $speed(i) \geq 0$

then

act1 : $speed(i) := speed(i) + e * acc(i)$
act2 : $position(i) := position(i) + e * speed(i) + acc(i) * (e * e/2)$
act3 : $phase(i) := 1$

end

END

3.2.4 m1

An Event-B Specification of m1
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE m1
REFINES m0
SEES c1
VARIABLES

phase
position
acc
speed
limit

INVARIANTS

inv3 : $\forall j \cdot j \in 1 \mathinner{..} c \Rightarrow position(j-1) \geq limit(j)$
inv1 : $\forall j \cdot j \in 1 \mathinner{..} c \Rightarrow position(j-1) \geq position(j)$

EVENTS

Initialisation

extended

begin

act2 : $phase := trains \times \{1\}$
act3 : $position := trains \times \{0\}$
act4 : $acc := trains \times \{0\}$
act5 : $speed := trains \times \{0\}$
act6 : $limit := trains \times \{0\}$

end

Event *get_limit_first* $\widehat{=}$

extends *get_limit*

any

x
i

where

grd1 : $x \in 1 \mathinner{..} 300$
grd2 : $i \in trains$
limit(i)− position(i)< 12,limit(i)=0
grd8 : $i = 0$
grd7 : $limit(i) - position(i) < 12$

then

act1 : $limit(i) := limit(i) + x$

end

Event *get_limit_following* $\widehat{=}$

refines *get_limit*

any

i

where

grd2 : $i \in trains$
grd3 : $\top$
limit(i)− position(i)< 12
grd4 : $\top$
limit(i)=0
grd5 : $i \in 1 \mathinner{..} c$

grd1 : $min(\{position(i-1) - limit(i), 300\}) \in 1 \mathinner{..} 300$

with

x : x=min({ position(i- 1)- limit(i),300})

then

act1 : $limit(i) := limit(i) + min(\{position(i-1) - limit(i), 300\})$

end

Event *decide_1* $\widehat{=}$

extends *decide_1*

any

i

where

grd5 : $i \in trains$

grd1 : $phase(i) = 1$

grd2 : $2 * limit(i) - 2 * position(i) < speed(i) * speed(i) + (A * e * e + 2 * e * speed(i)) * (A + b)$

grd3 : $A > 0$

grd4 : $speed(i) > 0$

then

act1 : $acc(i) := -b$

act2 : $phase(i) := 2$

end

Event *decide_2* $\widehat{=}$

extends *decide_2*

any

i

where

grd7 : $i \in trains$

grd1 : $phase(i) = 1$

grd2 : $2 * limit(i) - 2 * position(i) \geq speed(i) * speed(i) + (A * e * e + 2 * e * speed(i)) * (A + b)$

grd3 : $speed(i) \geq 0$

grd4 : $e > 0$

grd5 : $A > 0$

grd6 : $speed(i) + e * A \leq sl$

e=2,A=2

then

act1 : $acc(i) := A$

act2 : $phase(i) := 2$

end

Event *decide_3* $\widehat{=}$

extends *decide_3*

any

i

where

grd1 : $i \in trains$

grd2 : $phase(i) = 1$
grd3 : $2*limit(i)-2*position(i) \geq speed(i)*speed(i)+(A*e*e+2*e*speed(i))*(A+b)$
grd4 : $speed(i) > 0$
grd5 : $speed(i) + e * A > sl$

then

act1 : $acc(i) := 0$
act2 : $phase(i) := 2$

end

Event *drive1* $\widehat{=}$

extends *drive1*

any

i

where

grd1 : $i \in trains$
grd2 : $phase(i) = 2$
grd3 : $speed(i) + e * acc(i) \leq 0$
e=2

then

act1 : $speed(i) := 0$
act2 : $position(i) := position(i) + (speed(i) * speed(i))/(2 * b)$
act3 : $phase(i) := 1$

end

Event *drive2* $\widehat{=}$

extends *drive2*

any

i

where

grd1 : $i \in trains$
grd2 : $phase(i) = 2$
grd3 : $speed(i) + e * acc(i) > 0$
e=2
grd4 : $speed(i) \geq 0$

then

act1 : $speed(i) := speed(i) + e * acc(i)$
act2 : $position(i) := position(i) + e * speed(i) + acc(i) * (e * e/2)$
act3 : $phase(i) := 1$

end

END

Chapter 4

Aircraft Collision Avoidance

4.1 Context

4.1.1 c0

An Event-B Specification of f_c0
Creation Date: 1Nov2013 @ 04:45:41 PM

CONTEXT f_c0
CONSTANTS

$rhoi$
v
p
phi
r
$sqrt$
sin
phi_d_2

AXIOMS

axm0_1 : $rhoi > 0$
Inital distance of both aircrafts

axm0_2 : $v > 0$
axm0_3 : $p > 0$
minimal distance between aircrafts

axm0_4 : $phi > 0$
axm0_6 : $sqrt \in \mathbb{N} \rightarrow \mathbb{N}$
axm0_7 : $sin \in \mathbb{Z} \rightarrow -1 \mathinner{..} 1$
axm0_5 : $2 * rhoi * sin(phi\_d\_2) \geq p * sqrt(3)$
Initial distance greater than p*sqrt(3)

axm0_8 : $r > 0$
axm0_9 : $p \leq 2 * r * sin(phi\_d\_2)$

axm0_10 : $r * sqrt(3) \leq rhoi$

axm8 : $sqrt(3) > 1$

axm9 : $sqrt(3) * sqrt(3) = 3$

axm15 : $sin(phi\_d\_2) > 0$

axm16 : $phi\_d\_2 = phi/2$

axm20 : $2 * rhoi * sin(phi\_d\_2) \geq p$

END

4.1.2 c1

An Event-B Specification of f_c1
Creation Date: 1Nov2013 @ 04:45:41 PM

CONTEXT f_c1

EXTENDS f_c0

CONSTANTS

$t1$

$t2$

pi

$t3$

cos

AXIOMS

axm3 : $pi \in 3 \mathinner{..} 4$

axm1 : $t1 = (rhoi - sqrt(3) * r)/v$

axm2 : $t2 = (pi * r)/(3 * v)$

axm4 : $t3 = (2 * pi * r)/(3 * v)$

axm9 : $t1 \geq 0$

axm7 : $t2 \geq 0$

axm8 : $t3 \geq 0$

axm5 : $cos \in \mathbb{Z} \rightarrow -1 \mathinner{..} 1$

axm6 : $v * t1 = rhoi - sqrt(3) * r$

axm12 : $5 - 4 * cos(v * t2/r) \in dom(sqrt)$

axm13 : $5 - 4 * cos(pi/3 - v * t2/r) \in dom(sqrt)$

axm10 : $sqrt(5 - 4 * cos(pi/3 - v * t2/r)) = 1$

axm11 : $sqrt(5 - 4 * cos(v * t2/r)) = sqrt(3)$

axm14 : $\forall t \cdot t \in 0 \mathinner{..} t1 \Rightarrow 2 * (rhoi - v * t) * sin(phi\_d\_2) \geq p$

axm17 : $\forall t \cdot t \in t1 \mathinner{..} t1 + t2$
$\Rightarrow$
$(5 - 4 * cos(pi/3 - v * (t - t1)/r)) \in dom(sqrt)$

axm18 : $\forall t \cdot t \in t1 + t2 + t3 \mathinner{..} t1 + t2 + t3 + t2$
$\Rightarrow$
$5 - 4 * cos(v * (t - (t1 + t2 + t3))/r) \in dom(sqrt)$

axm15 : $\forall t \cdot t \in t1 \mathinner{..} t1 + t2$
$\Rightarrow$
$2 * r * sqrt(5 - 4 * cos(pi/3 - v * (t - t1)/r)) * sin(phi\_d\_2) \geq p$
axm16 : $\forall t \cdot t \in t1 + t2 + t3 \mathinner{..} t1 + t2 + t3 + t2$
$\Rightarrow$
$2 * r * sqrt(5 - 4 * cos(v * (t - (t1 + t2 + t3))/r)) * sin(phi\_d\_2) \geq p$

END

4.2 Machine

4.2.1 m0

An Event-B Specification of f_m0
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE f_m0
This is a simplified version.
We remove the event AGREE from the original model presented in the paper

SEES f_c0
VARIABLES

$phase$
rho
INVARIANTS

inv0_1 : $phase \in 1 \mathinner{..} 5$
inv0_2 : $rho > 0$
inv0_4 : $2 * rho * sin(phi\_d\_2) \geq p$ Distance always greater than or equal to p
inv1 : $phase = 3 \Rightarrow rho = r$

EVENTS
Initialisation
begin

act1 : $rho := rhoi$
act2 : $phase := 1$

end
Event *start* $\widehat{=}$
when

grd1 : $phase = 1$

then

act1 : $phase := 2$
act2 : $rho := sqrt(3) * r$

end
Event *enter* $\widehat{=}$
when

grd1 : $phase = 2$

then

act1 : $phase := 3$
act3 : $rho := r$

end

Event *cycle* $\widehat{=}$

when

grd1 : $phase = 3$

then

act1 : $phase := 4$

end

Event *leave* $\widehat{=}$

when

grd1 : $phase = 4$

then

act1 : $phase := 5$
act3 : $rho := r * sqrt(3)$

end

END

4.2.2 m1

An Event-B Specification of f_m1
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE f_m1
REFINES f_m0
SEES f_c1
VARIABLES

rho_c
phase
now

INVARIANTS

inv1_1 : $rho\_c \in \mathbb{N} \pfun \mathbb{Z}$
inv1_2 : $now \in dom(rho\_c)$
inv1_3 : $rho = rho\_c(now)$
inv1_4 : $\forall t \cdot t \in dom(rho\_c) \Rightarrow 2 * rho\_c(t) * sin(phi\_d\_2) \geq p$
inv4 : $phase = 1 \Rightarrow now = 0$
inv7 : $phase = 1 \Rightarrow rho\_c = \{0 \mapsto rhoi\}$
inv9 : $phase = 3 \Rightarrow dom(rho\_c) = now - (pi * r)/(3 * v) \upto now$
inv11 : $phase = 2 \Rightarrow now = t1$
inv12 : $phase = 3 \Rightarrow now = t1 + t2$
inv13 : $phase = 4 \Rightarrow now = t1 + t2 + t3$

EVENTS
Initialisation
begin

act5 : $phase := 1$
act8 : $now := 0$
act9 : $rho\_c := \{0 \mapsto rhoi\}$

end
Event *start* $\widehat{=}$
refines *start*
when

grd1 : $phase = 1$

then

act1 : $phase := 2$
act5 : $rho\_c := \lambda t \cdot t \in now \mathinner{..} now + t1 | rhoi - v * (t - now)$
act6 : $now := now + t1$

end
Event *enter* $\widehat{=}$
refines *enter*
when

grd1 : $phase = 2$

then

act1 : $phase := 3$
act7 : $now := now + t2$
act8 : $rho\_c := \lambda t \cdot t \in now \mathinner{..} now + t2 | r * sqrt(5 - 4 * cos(pi/3 - (v * (t - now)/r)))$

end
Event *cycle* $\widehat{=}$
refines *cycle*
when

grd1 : $phase = 3$
grd2 : $v > 0$
grd3 : $r > 0$
grd4 : $v > 0$

then

act1 : $phase := 4$
act5 : $now := now + t3$
act6 : $rho\_c := \lambda t \cdot t \in now \mathinner{..} now + t3 | r$

end
Event *leave* $\widehat{=}$
refines *leave*
when

grd1 : $phase = 4$

then

act1 : $phase := 5$

act7 : $now := now + t2$

act8 : $rho\_c := \lambda t \cdot t \in now \mathinner{..} now + t2 | r * sqrt(5 - 4 * cos(v * (t - now)/r))$

end

END

Chapter 5

Water Tank

5.1 Context

5.1.1 c0

An Event-B Specification of w_c0
Creation Date: 1Nov2013 @ 04:45:41 PM

CONTEXT w_c0

SETS

P

CONSTANTS

on
off
LM
Lm
$v1$
$v2$
t_sen
t_act
Li
$PUMPi$

AXIOMS

axm1 : $partition(P, \{on\}, \{off\})$
axm2 : $LM > 0$
axm3 : $Lm > 0$
axm4 : $v1 > 0$
axm5 : $v2 > 0$
axm6 : $t\_sen > 0$
axm7 : $t\_act > 0$
axm8 : $LM - v1 * t\_sen \geq Lm + v2 * t\_sen$
axm9 : $t\_sen > t\_act$

axm10 : $Lm < LM$

thm1 : $Lm + v2 * t\_act < LM - v1 * t\_act$

thm2 : $v1 * t\_act - v2 * t\_sen + v2 * t\_act \leq v1 * t\_act$

thm3 : $-v2 * t\_act + v1 * t\_sen - v1 * t\_act \geq -v2 * t\_act$

axm11 : $PUMPi = on \Rightarrow Lm \leq Li$

axm16 : $PUMPi = off \Rightarrow Li \leq LM$

axm12 : $PUMPi = on \Rightarrow Li \leq LM - v1 * t\_act$

axm13 : $PUMPi = off \Rightarrow Lm + v2 * t\_act \leq Li$

axm14 : $PUMPi = on \Rightarrow Li \in Lm \mathinner{..} LM$

axm15 : $PUMPi = off \Rightarrow Li \in Lm \mathinner{..} LM$

END

5.2 Machine

5.2.1 m0

An Event-B Specification of w_m0
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE w_m0

In this abstraction, we suppose that
the two mechanisms:
(1) physical pump possible modification
(2) sending of the water level
occur simultaneously.
In the next refinement, we shall separate
these two mechanisms.

SEES w_c0

VARIABLES

$PUMP$

$pump$

L

$phase$

INVARIANTS

inv0_1 : $PUMP \in P$

inv0_2 : $pump \in P$

inv0_3 : $phase \in \{1, 2\}$

inv0_4 : $phase = 1 \Rightarrow pump = PUMP$

When in the controller, the status of the physical
PUMP agrees with the pump command that has been sent
previously.

inv0_5 : $L \in Lm \mathinner{..} LM$

The safety invariant

inv0_10 : $PUMP = on \Rightarrow L \leq LM - v1 * t\_act$
When the physical PUMP is working then the water level is always smaller than or equal to tM minus the quantity corresponding to the delay t_act (with speed v1).

inv0_11 : $PUMP = off \Rightarrow L \geq Lm + v2 * t\_act$
When the physical PUMP is stopped then the water level is always greater than or equal to tm plus the quantity corresponding to the delay t_act (with speed v2).

inv0_12 :
$phase = 2 \wedge$
$PUMP = on \wedge$
$pump = on$
$\Rightarrow$
$L + v1 * t\_sen \leq LM - v1 * t\_act$

inv0_13 : $phase = 2 \wedge$
$PUMP = off \wedge$
$pump = off$
$\Rightarrow$
$L - v2 * t\_sen \geq Lm + v2 * t\_act$

inv0_14 : $phase = 2 \wedge$
$PUMP = on \wedge$
$pump = off$
$\Rightarrow$
$Lm \leq L + v1 * t\_act - v2 * t\_sen$

inv0_15 : $phase = 2 \wedge$
$PUMP = off \wedge$
$pump = on$
$\Rightarrow$
$L - v2 * t\_act + v1 * t\_sen \leq LM$

thm0_1 : $L - v2 * t\_act + v1 * t\_sen - v1 * t\_act < L - v2 * t\_act + v1 * t\_sen$

thm0_2 : $L + v1 * t\_act - v2 * t\_sen < L + v1 * t\_act - v2 * t\_sen + v2 * t\_act$

thm0_3 : $phase = 2 \wedge$
$PUMP = on \wedge$
$pump = on$
$\Rightarrow$
$L + v1 * t\_sen \leq LM$
(inv0_6 in paper)

thm0_4 : $phase = 2 \wedge$
$PUMP = off \wedge$
$pump = off$
$\Rightarrow$
$Lm \leq L - v2 * t\_sen$
(inv0_7 in paper)

thm0_5 : $phase = 2 \wedge$
$PUMP = off \wedge$
$pump = on$
$\Rightarrow$
$L - v2 * t\_act + v1 * t\_sen - v1 * t\_act \leq LM$
(inv0_8 in paper)

thm0_6 : $phase = 2 \wedge$
$PUMP = on \wedge$
$pump = off$
$\Rightarrow$
$Lm \leq L + v1 * t\_act - v2 * t\_sen + v2 * t\_act$
(inv0_9 in paper)

EVENTS

Initialisation

begin

act1 : $PUMP := PUMPi$
act2 : $pump := PUMPi$
act3 : $L := Li$
act4 : $phase := 1$

end

Event *decide_1* $\widehat{=}$
Do nothing. The pump should be still on.

when

grd1 : $phase = 1$
grd2 : $pump = on$
grd3 : $L + v1 * t\_sen \leq LM - v1 * t\_act$
The water level can continue to go up.
It will not be dangerous at next step in
t_sen seconds.

then

act1 : $phase := 2$

end

Event *decide_2* $\widehat{=}$
Send the command to stop the pump.

when

grd1 : $phase = 1$
grd2 : $pump = on$
grd3 : $L + v1 * t\_sen > LM - v1 * t\_act$
The water level cannot continue to go up.
It will be dangerous (too high) at next step
in t_sen seconds.
thm1 : $L \leq LM - v1 * t\_act$
thm2 : $LM - v1 * t\_sen \geq Lm + v2 * t\_sen$

then

act1 : $pump := off$
act2 : $phase := 2$

end

Event *decide_3* $\widehat{=}$

Do nothing. The pump should be still off.

when

grd1 : $phase = 1$
grd2 : $pump = off$
grd3 : $L - v2 * t\_sen \geq Lm + v2 * t\_act$
The water level can continue to go down.
It will not be dangerous at next step in
t_sen seconds.

then

act1 : $phase := 2$

end

Event *decide_4* $\widehat{=}$

Send the command to start the pump.

when

grd1 : $phase = 1$
grd2 : $pump = off$
grd3 : $L - v2 * t\_sen < Lm + v2 * t\_act$
The water level cannot continue to go down.
It will be dangerous (too low) at next step
in t_sen seconds.
thm1 : $L \geq Lm + v2 * t\_act$
thm2 : $LM - v1 * t\_sen \geq Lm + v2 * t\_sen$

then

act1 : $pump := on$
act2 : $phase := 2$

end

Event *env_1* $\widehat{=}$

The pump state (on) is not changed. Send the water level

when

grd1 : $phase = 2$
grd2 : $PUMP = on$
grd3 : $pump = on$

then

act1 : $L := L + v1 * t\_sen$
act2 : $phase := 1$

end

Event *env_2* $\widehat{=}$

The pump state (off) is not changed. Send the water level

when

grd1 : $phase = 2$
grd2 : $PUMP = off$
grd3 : $pump = off$

then

act1 : $L := L - v2 * t\_sen$
act2 : $phase := 1$

end

Event *env_3* $\widehat{=}$
The pump is stopped. Send the water level

when

grd1 : $phase = 2$
grd2 : $PUMP = on$
grd3 : $pump = off$
The pump receives the command to stop.
thm1 : $L + v1 * t\_act \leq LM$
thm2 : $v1 * t\_act - v2 * t\_sen + v2 * t\_act \leq v1 * t\_act$
thm3 : $L + v1 * t\_act - v2 * t\_sen + v2 * t\_act \leq L + v1 * t\_act$

then

act1 : $L := L + v1 * t\_act - v2 * t\_sen + v2 * t\_act$
The water level continues to go up with speed v1 for a time t_act. Then it goes down with speed v2 for a time t_sen-t_act.
act2 : $phase := 1$
act3 : $PUMP := off$

end

Event *env_4* $\widehat{=}$
The pump is started. Send the water level.

when

grd1 : $phase = 2$
grd2 : $PUMP = off$
grd3 : $pump = on$
The pump receives the command to start.
thm1 : $Lm \leq L - v2 * t\_act$
thm2 : $-v2 * t\_act + v1 * t\_sen - v1 * t\_act \geq -v2 * t\_act$
thm3 : $L - v2 * t\_act \leq L - v2 * t\_act + v1 * t\_sen - v1 * t\_act$
thm4 : $Lm \leq L - v2 * t\_act + v1 * t\_sen - v1 * t\_act$

then

act2 : $L := L - v2 * t\_act + v1 * t\_sen - v1 * t\_act$
The water level continues to go down with speed v2 for a time t_act. Then it goes up with speed v1 for a time t_sen-t_act.
act3 : $phase := 1$
act1 : $PUMP := on$

end

END

5.2.2 m1

An Event-B Specification of w_m1
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE w_m1

In this refinement, we separate the modification of the physical pump and the sending of the water level

REFINES w_m0

SEES w_c0

VARIABLES

$pump$

$phase$

LP

$PUMPP$

p_done

$change$

INVARIANTS

inv1_1 : $LP \in Lm \mathinner{..} LM$

inv1_2 : $PUMPP \in P$

inv1_3 : $p\_done = TRUE \Rightarrow PUMPP = pump$

inv1_4 : $phase = 1 \Rightarrow p\_done = FALSE$

inv1_5 : $p\_done = FALSE \Rightarrow LP = L$
phase=2 $\wedge$

inv1_6 : $p\_done = FALSE \Rightarrow PUMPP = PUMP$
phase=2 $\wedge$

inv1_7 : $p\_done = TRUE \wedge change = FALSE \Rightarrow PUMPP = PUMP$

inv1_8 : $p\_done = TRUE \wedge change = TRUE \Rightarrow PUMPP \neq PUMP$

inv1_9 : $p\_done = TRUE \wedge change = FALSE \wedge PUMPP = on \Rightarrow LP = L + v1 * t\_act$

inv1_10 : $p\_done = TRUE \wedge change = FALSE \wedge PUMPP = off \Rightarrow LP = L - v2 * t\_act$

inv1_11 : $p\_done = TRUE \wedge change = TRUE \wedge PUMPP = on \Rightarrow LP = L - v2 * t\_act$

inv1_12 : $p\_done = TRUE \wedge change = TRUE \wedge PUMPP = off \Rightarrow LP = L + v1 * t\_act$

EVENTS

Initialisation

begin

act2 : $pump := PUMPi$

act4 : $phase := 1$

act5 : $LP := Li$

act6 : $PUMPP := PUMPi$

act7 : $p\_done := FALSE$

act8 : $change := FALSE$

end

Event *decide_1* $\widehat{=}$

Same as in abstraction

refines *decide_1*

when

grd1 : $phase = 1$

grd2 : $pump = on$

grd3 : $LP + v1 * t\_sen \leq LM - v1 * t\_act$

then

act1 : $phase := 2$

end

Event *decide_2* $\widehat{=}$

Same as in abstraction

refines *decide_2*

when

grd1 : $phase = 1$
grd2 : $pump = on$
grd3 : $LP + v1 * t\_sen > LM - v1 * t\_act$

The water level cannot continue to go up. It will be dangerous (too high) at next step in t_sen seconds.

then

act1 : $pump := off$
act2 : $phase := 2$

end

Event *decide_3* $\widehat{=}$

Same as in abstraction

refines *decide_3*

when

grd1 : $phase = 1$
grd2 : $pump = off$
grd3 : $LP - v2 * t\_sen \geq Lm + v2 * t\_act$

The water level can continue to go down. It will not be dangerous at next step in t_sen seconds.

then

act1 : $phase := 2$

end

Event *decide_4* $\widehat{=}$

Same as in abstraction

refines *decide_4*

when

grd1 : $phase = 1$
grd2 : $pump = off$
grd3 : $LP - v2 * t\_sen < Lm + v2 * t\_act$

The water level cannot continue to go down. It will be dangerous (too low) at next step in t_sen seconds.

then

act1 : $pump := on$
act2 : $phase := 2$

end

Event *pump_off* $\widehat{=}$

Possible modification of the pump when it is off

when

grd1 : $phase = 2$
grd2 : $PUMPP = off$
grd3 : $p\_done = FALSE$

then

act1 : $LP := LP - v2 * t\_act$
Water level continues to go down at speed v2 for t_act seconds
act2 : $PUMPP := pump$
act3 : $p\_done := TRUE$
act4 : $change := bool(pump = on)$

end

Event *pump_on* $\widehat{=}$
Possible modification of the pump when it is on

when

grd1 : $phase = 2$
grd2 : $PUMPP = on$
grd3 : $p\_done = FALSE$

then

act1 : $LP := LP + v1 * t\_act$
Water level continues to go up at speed v1 for t_act seconds
act2 : $PUMPP := pump$
act3 : $p\_done := TRUE$
act4 : $change := bool(pump = off)$

end

Event *env_1* $\widehat{=}$
Sending the water level when pump is on and no change
refines *env_1*

when

grd1 : $phase = 2$
grd2 : $PUMPP = on$
grd4 : $p\_done = TRUE$
grd5 : $change = FALSE$

then

act1 : $LP := LP + v1 * t\_sen - v1 * t\_act$
act2 : $phase := 1$
act3 : $p\_done := FALSE$

end

Event *env_2* $\widehat{=}$
Sending the water level when pump is off and no change
refines *env_2*

when

grd1 : $phase = 2$
grd2 : $PUMPP = off$
grd3 : $p\_done = TRUE$

grd4 : $change = FALSE$

then

act1 : $LP := LP - v2 * t\_sen + v2 * t\_act$
act2 : $phase := 1$
act3 : $p\_done := FALSE$

end

Event *env_3* $\widehat{=}$
Sending the water level when pump is off and change
refines *env_3*

when

grd1 : $phase = 2$
grd2 : $PUMPP = off$
grd3 : $p\_done = TRUE$
grd4 : $change = TRUE$

then

act1 : $LP := LP - v2 * t\_sen + v2 * t\_act$
act2 : $phase := 1$
act3 : $p\_done := FALSE$

end

Event *env_4* $\widehat{=}$
Sending the water level when pump is on and change
refines *env_4*

when

grd1 : $phase = 2$
grd2 : $PUMPP = on$
grd3 : $p\_done = TRUE$
grd4 : $change = TRUE$

then

act1 : $LP := LP + v1 * t\_sen - v1 * t\_act$
act2 : $phase := 1$
act3 : $p\_done := FALSE$

end

END

5.2.3 m2

An Event-B Specification of w_m2
Creation Date: 1Nov2013 @ 04:45:41 PM

MACHINE w_m2
Technical refinement for merging env1 and env4
and also env2 and env3
REFINES w_m1
SEES w_c0
VARIABLES

pump

$phase$

LP

$PUMPP$

p_done

EVENTS

Initialisation

begin

act2 : $pump := PUMPi$

act4 : $phase := 1$

act5 : $LP := Li$

act6 : $PUMPP := PUMPi$

act7 : $p\_done := FALSE$

end

Event *decide_1* $\widehat{=}$

Same as in abstraction

extends *decide_1*

when

grd1 : $phase = 1$

grd2 : $pump = on$

grd3 : $LP + v1 * t\_sen \leq LM - v1 * t\_act$

then

act1 : $phase := 2$

end

Event *decide_2* $\widehat{=}$

Same as in abstraction

extends *decide_2*

when

grd1 : $phase = 1$

grd2 : $pump = on$

grd3 : $LP + v1 * t\_sen > LM - v1 * t\_act$

The water level cannot continue to go up.
It will be dangerous (too high) at next step
in t_sen seconds.

then

act1 : $pump := off$

act2 : $phase := 2$

end

Event *decide_3* $\widehat{=}$

Same as in abstraction

extends *decide_3*

when

grd1 : $phase = 1$

grd2 : $pump = off$

grd3 : $LP - v2 * t\_sen \geq Lm + v2 * t\_act$
The water level can continue to go down.
It will not be dangerous at next step in
t_sen seconds.

then

act1 : $phase := 2$

end

Event *decide_4* $\widehat{=}$
Same as in abstraction
extends *decide_4*

when

grd1 : $phase = 1$
grd2 : $pump = off$
grd3 : $LP - v2 * t\_sen < Lm + v2 * t\_act$
The water level cannot continue to go down.
It will be dangerous (too low) at next step
in t_sen seconds.

then

act1 : $pump := on$
act2 : $phase := 2$

end

Event *pump_on* $\widehat{=}$
refines *pump_on*

when

grd1 : $phase = 2$
grd2 : $PUMPP = on$
grd3 : $p\_done = FALSE$

then

act1 : $LP := LP + v1 * t\_act$
act2 : $PUMPP := pump$
act3 : $p\_done := TRUE$

end

Event *pump_off* $\widehat{=}$
refines *pump_off*

when

grd1 : $phase = 2$
grd2 : $PUMPP = off$
grd3 : $p\_done = FALSE$

then

act1 : $LP := LP - v2 * t\_act$
act2 : $PUMPP := pump$
act3 : $p\_done := TRUE$

end

Event *sensor_on* $\widehat{=}$
refines *env_1, env_4*

when

grd1 : $phase = 2$
grd2 : $PUMPP = on$
grd3 : $p\_done = TRUE$

then

act1 : $LP := LP + v1 * t\_sen - v1 * t\_act$
act2 : $phase := 1$
act3 : $p\_done := FALSE$

end

Event *sensor_off* $\widehat{=}$
refines *env_2, env_3*

when

grd1 : $phase = 2$
grd2 : $PUMPP = off$
grd3 : $p\_done = TRUE$

then

act1 : $LP := LP - v2 * t\_sen + v2 * t\_act$
act2 : $phase := 1$
act3 : $p\_done := FALSE$

end

END