Contents

CONTEXT c_basal

SETS

BASALMODE

CONSTANTS

basal_max

c

suspended

delivering

stop

AXIOMS

axm1: $basal\_max \in \mathbb{N}_1$

axm2: $c = 48$

axm3: $partition(BASALMODE, \{suspended\}, \{delivering\}, \{stop\})$

END

CONTEXT c_basal2
EXTENDS c_basal
SETS
PROG0
CONSTANTS
null
call_get_min
return_get_min
call_get_max
return_get_max
PROC_BASAL
AXIOMS
axm2: $PROC\_BASAL \subseteq PROG0$
axm1: $partition(PROC\_BASAL, \{null\}, \{call\_get\_min\}, \{return\_get\_min\}, \{call\_get\_max\}, \{return\_get\_max\})$

END

CONTEXT c_prog
EXTENDS c_basal2
CONSTANTS

PROG
call_basal_start
return_basal_start
call_basal_stop
return_basal_stop
call_basal_suspend
return_basal_suspend
call_basal_resume
return_basal_resume
call_basal_update
return_basal_update
call_normal_suspend
return_normal_suspend
call_normal_finish
return_normal_finish
call_normal_resume
return_normal_resume
call_normal_start
return_normal_start

AXIOMS

axm2: $PROG \subseteq PROG0$

axm1:
$partition(PROG, \{null\}, \{call\_basal\_start\}, \{return\_basal\_start\}, \{call\_basal\_stop\}, \{return\_basal\_stop\}, \{call\_basal\_suspend\}, \{return\_basal\_suspend\}, \{call\_basal\_resume\}, \{return\_basal\_resume\}, \{call\_basal\_update\}, \{return\_basal\_update\}, \{call\_normal\_start\}, \{return\_normal\_start\}, \{call\_normal\_suspend\}, \{return\_normal\_suspend\}, \{call\_normal\_finish\}, \{return\_normal\_finish\}, \{call\_normal\_resume\}, \{return\_normal\_resume\})$

END

CONTEXT c_prog2
EXTENDS c_prog
CONSTANTS

call_sd_start_s
return_sd_start_s
call_sd_start_d
return_sd_start_d
call_sd_update
return_sd_update
call_sd_finish
return_sd_finish
call_sd_suspend
return_sd_suspend
call_sd_resume
return_sd_resume
call_sd_preempt
return_sd_preempt
call_sd_resume_preempt
return_sd_resume_preempt
pg2

AXIOMS

axm2: $pg2 \subseteq PROG0$
axm1: $partition(pg2, \{call\_sd\_start\_s\}, \{return\_sd\_start\_s\}, \{call\_sd\_start\_d\}, \{return\_sd\_start\_d\}, \{call\_sd\_update\}, \{return\_sd\_update\}, \{call\_sd\_finish\}, \{return\_sd\_finish\}, \{call\_sd\_suspend\}, \{return\_sd\_suspend\}, \{call\_sd\_resume\}, \{return\_sd\_resume\}, \{call\_sd\_preempt\}, \{return\_sd\_preempt\}, \{call\_sd\_resume\_preempt\}, \{return\_sd\_resume\_preempt\})$
axm3: $PROG \cap pg2 = \varnothing$

END

CONTEXT c_prog2_anim
EXTENDS c_prog2
AXIOMS
axm1: $PROG0 = PROG \cup pg2 \cup PROC\_BASAL$
END

CONTEXT c_sd_bolus

SETS

SD

SDF

CONSTANTS

deliver

off

suspend

preempt

s

d

AXIOMS

axm1: $partition(SD, \{deliver\}, \{off\}, \{suspend\}, \{preempt\})$

axm2: $partition(SDF, \{s\}, \{d\})$

END

CONTEXT c_normalbolus
CONSTANTS
normal_bolus_rate
AXIOMS
axm1: $normal\_bolus\_rate > 0$
END

CONTEXT c_normalbolus_anim
EXTENDS c_normalbolus
AXIOMS
axm1: $normal\_bolus\_rate = 2$
END

MACHINE control

VARIABLES

normal_bolus_work

sd_bolus_work

sd_preempted_by_normal

INVARIANTS

inv1: $normal\_bolus\_work \in BOOL$

inv2: $sd\_bolus\_work \in BOOL$

inv3: $sd\_preempted\_by\_normal \in BOOL$

inv4: $normal\_bolus\_work = TRUE \Rightarrow (sd\_bolus\_work = FALSE \lor sd\_preempted\_by\_normal = TRUE)$

inv5: $sd\_preempted\_by\_normal = TRUE \Rightarrow sd\_bolus\_work = TRUE$

EVENTS

Initialisation

begin

act1: $normal\_bolus\_work := FALSE$

act2: $sd\_bolus\_work := FALSE$

act3: $sd\_preempted\_by\_normal := FALSE$

end

Event normal_bolus_start_1 ⟨ordinary⟩ $\widehat{=}$

when

grd1: $normal\_bolus\_work = FALSE$

grd2: $sd\_bolus\_work = TRUE$

grd3: $sd\_preempted\_by\_normal = FALSE$

then

act1: $normal\_bolus\_work := TRUE$

act2: $sd\_preempted\_by\_normal := TRUE$

end

Event normal_bolus_start_2 ⟨ordinary⟩ $\widehat{=}$

when

grd1: $normal\_bolus\_work = FALSE$

grd2: $sd\_bolus\_work = FALSE$

then

act1: $normal\_bolus\_work := TRUE$

end

Event normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$

when

grd1: $normal\_bolus\_work = TRUE$

then

act1: $normal\_bolus\_work := FALSE$

end

Event square_or_dual_bolus_start ⟨ordinary⟩ $\widehat{=}$

when

grd1: $sd\_bolus\_work = FALSE$

grd2: $normal\_bolus\_work = FALSE$

then

act1: $sd\_bolus\_work := TRUE$

end

Event square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$

when

grd1: $sd\_bolus\_work = TRUE$

grd2: $sd\_preempted\_by\_normal = FALSE$

then

act1: $sd\_bolus\_work := FALSE$

end

Event square_or_dual_bolus_resume_of_normal ⟨ordinary⟩ $\widehat{=}$

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$

then

act1: $sd\_preempted\_by\_normal := FALSE$

end

END

MACHINE control2
REFINES control
VARIABLES
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
INVARIANTS
inv2_8: $sd\_suspend \in BOOL$
inv3_9: $normal\_suspend \in BOOL$
inv4_10: $sd\_suspend = TRUE \Rightarrow sd\_bolus\_work = TRUE \wedge sd\_preempted\_by\_normal = FALSE$
inv5_11: $normal\_suspend = TRUE \Rightarrow normal\_bolus\_work = TRUE$
EVENTS
Initialisation
begin
act1: $normal\_bolus\_work := FALSE$
act2: $sd\_bolus\_work := FALSE$
act3: $sd\_preempted\_by\_normal := FALSE$
act7: $sd\_suspend := FALSE$
act8: $normal\_suspend := FALSE$
end
Event normal_bolus_start_1 ⟨ordinary⟩ $\widehat{=}$
refines normal_bolus_start_1
when
grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $normal\_bolus\_work := TRUE$
act2: $sd\_preempted\_by\_normal := TRUE$
end
Event normal_bolus_start_2 ⟨ordinary⟩ $\widehat{=}$
refines normal_bolus_start_2
when
grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_suspend = FALSE$
then
act1: $normal\_bolus\_work := TRUE$
end
Event normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$
refines normal_bolus_finish
when
grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
then
act1: $normal\_bolus\_work := FALSE$
end
Event square_or_dual_bolus_start ⟨ordinary⟩ $\widehat{=}$
refines square_or_dual_bolus_start
when
grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$

then
act1: $sd\_bolus\_work := TRUE$
end
Event square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$
refines square_or_dual_bolus_finish
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $sd\_bolus\_work := FALSE$
end
Event square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$
refines square_or_dual_bolus_resume_of_normal
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $sd\_preempted\_by\_normal := FALSE$
end
Event normal_suspend ⟨ordinary⟩ $\widehat{=}$
when
grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
then
act1: $normal\_suspend := TRUE$
end
Event normal_resume ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_finish
when
grd1: $normal\_bolus\_work = TRUE$
grd2: $normal\_suspend = TRUE$
then
act1: $normal\_bolus\_work := FALSE$
act2: $normal\_suspend := FALSE$
end
Event sd_suspend ⟨ordinary⟩ $\widehat{=}$
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $sd\_suspend := TRUE$
end
Event sd_resume ⟨ordinary⟩ $\widehat{=}$
refines square_or_dual_bolus_finish
when
grd1: $sd\_suspend = TRUE$
then
act1: $sd\_bolus\_work := FALSE$
act2: $sd\_suspend := FALSE$
end
END

MACHINE control3
REFINES control2
VARIABLES
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
INVARIANTS
inv1_12: $basal\_work \in BOOL$
inv2_13: $basal\_suspend \in BOOL$
inv3_14: $basal\_suspend = TRUE \Rightarrow basal\_work = TRUE$
EVENTS
Initialisation
begin
act1: $normal\_bolus\_work := FALSE$
act2: $sd\_bolus\_work := FALSE$
act3: $sd\_preempted\_by\_normal := FALSE$
act7: $sd\_suspend := FALSE$
act8: $normal\_suspend := FALSE$
act9: $basal\_work := FALSE$
act10: $basal\_suspend := FALSE$
end
Event normal_bolus_start_1 ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_start_1
when
grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $normal\_bolus\_work := TRUE$
act2: $sd\_preempted\_by\_normal := TRUE$
end
Event normal_bolus_start_2 ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_start_2
when
grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_suspend = FALSE$
then
act1: $normal\_bolus\_work := TRUE$
end
Event normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_finish
when
grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
then
act1: $normal\_bolus\_work := FALSE$
end
Event square_or_dual_bolus_start ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_start

when
grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
then
act1: $sd\_bolus\_work := TRUE$
end
Event square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_finish
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $sd\_bolus\_work := FALSE$
end
Event square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_resume_from_normal
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $sd\_preempted\_by\_normal := FALSE$
end
Event normal_suspend ⟨ordinary⟩ $\widehat{=}$
extends normal_suspend
when
grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
then
act1: $normal\_suspend := TRUE$
end
Event sd_suspend ⟨ordinary⟩ $\widehat{=}$
extends sd_suspend
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $sd\_suspend := TRUE$
end
Event normal_resume ⟨ordinary⟩ $\widehat{=}$
extends normal_resume
when
grd1: $normal\_bolus\_work = TRUE$
grd2: $normal\_suspend = TRUE$
then
act1: $normal\_bolus\_work := FALSE$
act2: $normal\_suspend := FALSE$
end
Event sd_resume ⟨ordinary⟩ $\widehat{=}$
extends sd_resume
when
grd1: $sd\_suspend = TRUE$
then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_suspend := FALSE$

end

Event basal_start ⟨ordinary⟩ $\widehat{=}$

when

grd1: $basal\_work = FALSE$
grd3: $basal\_suspend = FALSE$

then

act1: $basal\_work := TRUE$

end

Event basal_stop ⟨ordinary⟩ $\widehat{=}$

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

then

act1: $basal\_work := FALSE$

end

Event basal_suspend ⟨ordinary⟩ $\widehat{=}$

when

grd1: $basal\_work = TRUE$
grd3: $basal\_suspend = FALSE$

then

act1: $basal\_suspend := TRUE$

end

Event basal_resume ⟨ordinary⟩ $\widehat{=}$

when

grd1: $basal\_suspend = TRUE$

then

act1: $basal\_suspend := FALSE$

end

END

MACHINE control4
REFINES control3
VARIABLES

normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate

INVARIANTS

inv1_15: $pump\_rate \in \mathbb{N}$
inv2_16: $basal\_rate \in \mathbb{N}$
inv3_17: $normal\_rate \in \mathbb{N}$
inv4_18: $sd\_rate \in \mathbb{N}$
inv8_19: $basal\_suspend = TRUE \Rightarrow basal\_rate = 0$
inv9_20: $normal\_suspend = TRUE \Rightarrow normal\_rate = 0$
inv10_21: $sd\_suspend = TRUE \Rightarrow sd\_rate = 0$
inv11_22: $sd\_rate = 0 \lor normal\_rate = 0$
inv12_23: $sd\_rate \neq 0 \Rightarrow (sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE)$
inv13_24: $normal\_rate \neq 0 \Rightarrow normal\_bolus\_work = TRUE$
inv14_25: $pump\_rate = normal\_rate + sd\_rate + basal\_rate$

EVENTS

Initialisation ⟨extended⟩
begin
act1: $normal\_bolus\_work := FALSE$
act2: $sd\_bolus\_work := FALSE$
act3: $sd\_preempted\_by\_normal := FALSE$
act7: $sd\_suspend := FALSE$
act8: $normal\_suspend := FALSE$
act9: $basal\_work := FALSE$
act10: $basal\_suspend := FALSE$
act11: $pump\_rate := 0$
act12: $basal\_rate := 0$
act13: $normal\_rate := 0$
act14: $sd\_rate := 0$
end

Event normal_bolus_start_1 ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_start_1
any
r
where
grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}$
then
act1: $normal\_bolus\_work := TRUE$
act2: $sd\_preempted\_by\_normal := TRUE$
act3: $normal\_rate := r$

act4: $sd\_rate := 0$
act5: $pump\_rate := r + basal\_rate$

end

Event normal_bolus_start_2 ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_start_2

any
r
where
grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
then
act1: $normal\_bolus\_work := TRUE$
act2: $normal\_rate := r$
act3: $pump\_rate := r + basal\_rate$
end

Event normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_finish

when
grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
then
act1: $normal\_bolus\_work := FALSE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
end

Event square_or_dual_bolus_start ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_start

any
r
where
grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd4: $r \in \mathbb{N}_1$
then
act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
end

Event square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_finish

when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
end

Event square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_resume_from_normal

any
r
where

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}_1$

then

act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$

end

Event normal_suspend ⟨ordinary⟩ $\widehat{=}$

extends normal_suspend

when

grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$

then

act1: $normal\_suspend := TRUE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$

end

Event sd_suspend ⟨ordinary⟩ $\widehat{=}$

extends sd_suspend

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $sd\_suspend := TRUE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$

end

Event square_or_dual_update_rate ⟨ordinary⟩ $\widehat{=}$

any

r

where

grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $r \in \mathbb{N}_1$

then

act1: $sd\_rate := r$
act2: $pump\_rate := r + basal\_rate$

end

Event normal_resume ⟨ordinary⟩ $\widehat{=}$

extends normal_resume

when

grd1: $normal\_bolus\_work = TRUE$
grd2: $normal\_suspend = TRUE$

then

act1: $normal\_bolus\_work := FALSE$
act2: $normal\_suspend := FALSE$
act4: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$

end

Event sd_resume ⟨ordinary⟩ $\widehat{=}$

extends sd_resume

when

grd1: $sd\_suspend = TRUE$

then
act1: $sd\_bolus\_work := FALSE$
act2: $sd\_suspend := FALSE$
act3: $sd\_rate := 0$
act4: $pump\_rate := basal\_rate$
end
Event basal_start ⟨ordinary⟩ $\widehat{=}$
extends basal_start
any
r
where
grd1: $basal\_work = FALSE$
grd3: $basal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
then
act1: $basal\_work := TRUE$
act2: $basal\_rate := r$
act3: $pump\_rate := r + normal\_rate + sd\_rate$
end
Event basal_stop ⟨ordinary⟩ $\widehat{=}$
extends basal_stop
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
then
act1: $basal\_work := FALSE$
act2: $basal\_rate := 0$
act3: $pump\_rate := normal\_rate + sd\_rate$
end
Event basal_suspend ⟨ordinary⟩ $\widehat{=}$
extends basal_suspend
when
grd1: $basal\_work = TRUE$
grd3: $basal\_suspend = FALSE$
then
act1: $basal\_suspend := TRUE$
act2: $basal\_rate := 0$
act3: $pump\_rate := normal\_rate + sd\_rate$
end
Event basal_resume ⟨ordinary⟩ $\widehat{=}$
extends basal_resume
any
r
where
grd1: $basal\_suspend = TRUE$
grd2: $r \in \mathbb{N}$
then
act1: $basal\_suspend := FALSE$
act2: $basal\_rate := r$
act3: $pump\_rate := r + normal\_rate + sd\_rate$
end
Event basal_update_rate ⟨ordinary⟩ $\widehat{=}$
any
r
where
grd2: $basal\_suspend = FALSE$
grd3: $basal\_work = TRUE$
grd4: $r \in \mathbb{N}$

then

act1: $basal\_rate := r$

act2: $pump\_rate := r + normal\_rate + sd\_rate$

end

END

MACHINE control5
REFINES control4
VARIABLES

normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd

INVARIANTS

inv1_26: $time \in \mathbb{N}$
inv2_27: $t\_basal \in \mathbb{Z}$
inv3_28: $t\_normal \in \mathbb{Z}$
inv4_29: $t\_sd \in \mathbb{Z}$
inv5_30: $basal\_work = FALSE \Rightarrow t\_basal = 0$
inv6_31: $normal\_bolus\_work = FALSE \Rightarrow t\_normal = 0$
inv7_32: $sd\_bolus\_work = FALSE \Rightarrow t\_sd = 0$
inv1: $normal\_suspend = TRUE \Rightarrow normal\_bolus\_work = TRUE$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$
act2: $sd\_bolus\_work := FALSE$
act3: $sd\_preempted\_by\_normal := FALSE$
act7: $sd\_suspend := FALSE$
act8: $normal\_suspend := FALSE$
act9: $basal\_work := FALSE$
act10: $basal\_suspend := FALSE$
act11: $pump\_rate := 0$
act12: $basal\_rate := 0$
act13: $normal\_rate := 0$
act14: $sd\_rate := 0$
act15: $time := 0$
act16: $t\_basal := 0$
act17: $t\_normal := 0$
act18: $t\_sd := 0$

end

Event normal_bolus_start_1 ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_start_1

any

r
t

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$

grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}$
grd6: $t \in \mathbb{N}$

then

act1: $normal\_bolus\_work := TRUE$
act2: $sd\_preempted\_by\_normal := TRUE$
act3: $normal\_rate := r$
act4: $sd\_rate := 0$
act5: $pump\_rate := r + basal\_rate$
act6: $t\_normal := time + t$
act7: $t\_sd := t\_sd - time$

end

Event normal_bolus_start_2 ⟨ordinary⟩ $\widehat{=}$

extends normal_bolus_start_2

any

r
t

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
grd5: $t \in \mathbb{N}$

then

act1: $normal\_bolus\_work := TRUE$
act2: $normal\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_normal := time + t$

end

Event normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$

extends normal_bolus_finish

when

grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
grd4: $time = t\_normal$

then

act1: $normal\_bolus\_work := FALSE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_normal := 0$

end

Event square_or_dual_bolus_start ⟨ordinary⟩ $\widehat{=}$

extends square_or_dual_bolus_start

any

r
t

where

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd4: $r \in \mathbb{N}_1$
grd5: $t \in \mathbb{N}_1$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$

end

Event square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_finish
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$
then
act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
end
Event square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_resume_from_normal
any
r
where
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}_1$
then
act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t\_sd$
end
Event normal_suspend ⟨ordinary⟩ $\widehat{=}$
extends normal_suspend
when
grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
then
act1: $normal\_suspend := TRUE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_normal := t\_normal - time$
end
Event sd_suspend ⟨ordinary⟩ $\widehat{=}$
extends sd_suspend
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $sd\_suspend := TRUE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
end
Event square_or_dual_update_rate ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_update_rate
any
r
where
grd2: $sd\_suspend = FALSE$

grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $r \in \mathbb{N}_1$

then

act1: $sd\_rate := r$
act2: $pump\_rate := r + basal\_rate$

end

Event normal_resume ⟨ordinary⟩ $\widehat{=}$
refines normal_resume

when

grd1: $normal\_bolus\_work = TRUE$
grd2: $normal\_suspend = TRUE$

then

act1: $normal\_bolus\_work := FALSE$
act2: $normal\_suspend := FALSE$
act4: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
act5: $t\_normal := 0$

end

Event sd_resume ⟨ordinary⟩ $\widehat{=}$
extends sd_resume

when

grd1: $sd\_suspend = TRUE$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_suspend := FALSE$
act3: $sd\_rate := 0$
act4: $pump\_rate := basal\_rate$
act5: $t\_sd := 0$

end

Event basal_start ⟨ordinary⟩ $\widehat{=}$
extends basal_start

any

r
t

where

grd1: $basal\_work = FALSE$
grd3: $basal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
grd5: $t \in \mathbb{N}_1$

then

act1: $basal\_work := TRUE$
act2: $basal\_rate := r$
act3: $pump\_rate := r + normal\_rate + sd\_rate$
act4: $t\_basal := time + t$

end

Event basal_stop ⟨ordinary⟩ $\widehat{=}$
extends basal_stop

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

then

act1: $basal\_work := FALSE$
act2: $basal\_rate := 0$
act3: $pump\_rate := normal\_rate + sd\_rate$
act4: $t\_basal := 0$

end

Event basal_suspend ⟨ordinary⟩ $\widehat{=}$

extends basal_suspend

when

grd1: $basal\_work = TRUE$

grd3: $basal\_suspend = FALSE$

then

act1: $basal\_suspend := TRUE$

act2: $basal\_rate := 0$

act3: $pump\_rate := normal\_rate + sd\_rate$

act4: $t\_basal := 0$

end

Event basal_resume ⟨ordinary⟩ $\widehat{=}$

extends basal_resume

any

r

t

where

grd1: $basal\_suspend = TRUE$

grd2: $r \in \mathbb{N}$

grd3: $t \in \mathbb{N}_1$

then

act1: $basal\_suspend := FALSE$

act2: $basal\_rate := r$

act3: $pump\_rate := r + normal\_rate + sd\_rate$

act4: $t\_basal := t + time$

end

Event basal_update_rate ⟨ordinary⟩ $\widehat{=}$

extends basal_update_rate

any

r

t

where

grd2: $basal\_suspend = FALSE$

grd3: $basal\_work = TRUE$

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

grd5: $t\_basal = time$

grd6: $t \in \mathbb{N}_1$

then

act1: $basal\_rate := r$

act2: $pump\_rate := r + normal\_rate + sd\_rate$

act3: $t\_basal := time + t$

end

Event timer ⟨ordinary⟩ $\widehat{=}$

begin

act1: $time := time + 1$

end

END

MACHINE control6
REFINES control5
VARIABLES

normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
dmodule
d_update_time

INVARIANTS

inv1: $dmodule \in BOOL$
inv2: $d\_update\_time \in \mathbb{N}$
inv3: $dmodule = TRUE \Rightarrow sd\_bolus\_work = TRUE \land sd\_suspend = FALSE \land d\_update\_time > 0$
inv8: $basal\_work = TRUE \land basal\_suspend = FALSE \Rightarrow t\_basal \geq time$
inv9: $t\_basal \in \mathbb{Z}$
inv10: $t\_normal \in \mathbb{Z}$
inv14: $dmodule = TRUE \land sd\_preempted\_by\_normal = FALSE \land sd\_suspend = FALSE \Rightarrow d\_update\_time \geq time$
inv15: $dmodule = TRUE \land sd\_preempted\_by\_normal = TRUE \Rightarrow d\_update\_time \geq 0$
inv16: $dmodule = FALSE \Rightarrow d\_update\_time = 0$
inv17: $time = t\_sd \land sd\_preempted\_by\_normal = FALSE \Rightarrow dmodule = FALSE$
inv18: $time \geq d\_update\_time \land d\_update\_time = 0 \land sd\_preempted\_by\_normal = FALSE \Rightarrow dmodule = FALSE$
inv19: $time = d\_update\_time \land d\_update\_time \neq 0 \Rightarrow dmodule = TRUE$
inv20: $d\_update\_time \neq 0 \Rightarrow t\_sd > d\_update\_time$
inv21: $sd\_preempted\_by\_normal = TRUE \Rightarrow t\_sd \geq 0$
inv22: $sd\_preempted\_by\_normal = FALSE \land sd\_bolus\_work = TRUE \land sd\_suspend = FALSE \Rightarrow t\_sd \geq time$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$
act2: $sd\_bolus\_work := FALSE$
act3: $sd\_preempted\_by\_normal := FALSE$
act7: $sd\_suspend := FALSE$
act8: $normal\_suspend := FALSE$
act9: $basal\_work := FALSE$
act10: $basal\_suspend := FALSE$
act11: $pump\_rate := 0$
act12: $basal\_rate := 0$
act13: $normal\_rate := 0$
act14: $sd\_rate := 0$
act15: $time := 0$

act16: $t\_basal := 0$
act17: $t\_normal := 0$
act18: $t\_sd := 0$
act19: $dmodule := FALSE$
act20: $d\_update\_time := 0$

end

Event normal_bolus_start_1 ⟨ordinary⟩ $\widehat{=}$

extends normal_bolus_start_1

any

r
t
t2

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}$
grd6: $t \in \mathbb{N}$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \land time \neq d\_update\_time$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $normal\_bolus\_work := TRUE$
act2: $sd\_preempted\_by\_normal := TRUE$
act3: $normal\_rate := r$
act4: $sd\_rate := 0$
act5: $pump\_rate := r + basal\_rate$
act6: $t\_normal := time + t$
act7: $t\_sd := t\_sd - time$
act8: $d\_update\_time := t2$

end

Event normal_bolus_start_2 ⟨ordinary⟩ $\widehat{=}$

extends normal_bolus_start_2

any

r
t

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
grd5: $t \in \mathbb{N}$

then

act1: $normal\_bolus\_work := TRUE$
act2: $normal\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_normal := time + t$

end

Event normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$

extends normal_bolus_finish

when

grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
grd4: $time = t\_normal$

then

act1: $normal\_bolus\_work := FALSE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$

act4: $t\_normal := 0$

end

Event normal_suspend ⟨ordinary⟩ $\widehat{=}$

extends normal_suspend

when

grd1: $normal\_bolus\_work = TRUE$

grd3: $normal\_suspend = FALSE$

then

act1: $normal\_suspend := TRUE$

act2: $normal\_rate := 0$

act3: $pump\_rate := basal\_rate$

act4: $t\_normal := t\_normal - time$

end

Event normal_resume ⟨ordinary⟩ $\widehat{=}$

extends normal_resume

when

grd1: $normal\_bolus\_work = TRUE$

grd2: $normal\_suspend = TRUE$

grd3: $\top$

then

act1: $normal\_bolus\_work := FALSE$

act2: $normal\_suspend := FALSE$

act4: $normal\_rate := 0$

act3: $pump\_rate := basal\_rate$

act5: $t\_normal := 0$

end

Event square_or_dual_bolus_start_s ⟨ordinary⟩ $\widehat{=}$

extends square_or_dual_bolus_start

any

r

t

where

grd1: $sd\_bolus\_work = FALSE$

grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_suspend = FALSE$

grd4: $r \in \mathbb{N}_1$

grd5: $t \in \mathbb{N}_1$

then

act1: $sd\_bolus\_work := TRUE$

act2: $sd\_rate := r$

act3: $pump\_rate := r + basal\_rate$

act4: $t\_sd := time + t$

act5: $dmodule := FALSE$

end

Event square_or_dual_bolus_start_d ⟨ordinary⟩ $\widehat{=}$

extends square_or_dual_bolus_start

any

r

t

t0 t: both bolus, t0:for normal bolus

where

grd1: $sd\_bolus\_work = FALSE$

grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_suspend = FALSE$

grd4: $r \in \mathbb{N}_1$

grd5: $t \in \mathbb{N}_1$

grd6: $t0 \in \mathbb{N}_1$

grd7: $t > t0$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$
act5: $dmodule := TRUE$
act6: $d\_update\_time := time + t0$

end

Event square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_finish

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $dmodule := FALSE$

end

Event square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_resume_from_normal

any

r
t2

where

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd7: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $d\_update\_time := t2$

end

Event sd_suspend ⟨ordinary⟩ $\widehat{=}$
extends sd_suspend

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $sd\_suspend := TRUE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$
act6: $d\_update\_time := 0$

end

Event square_or_dual_update_rate ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_update_rate

any
r
where
grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE$
grd7: $time = d\_update\_time$
then
act1: $sd\_rate := r$
act2: $pump\_rate := r + basal\_rate$
act3: $dmodule := FALSE$
act4: $d\_update\_time := 0$
end
Event sd_resume ⟨ordinary⟩ $\widehat{=}$
extends sd_resume
when
grd1: $sd\_suspend = TRUE$
then
act1: $sd\_bolus\_work := FALSE$
act2: $sd\_suspend := FALSE$
act3: $sd\_rate := 0$
act4: $pump\_rate := basal\_rate$
act5: $t\_sd := 0$
end
Event basal_start ⟨ordinary⟩ $\widehat{=}$
extends basal_start
any
r
t
where
grd1: $basal\_work = FALSE$
grd3: $basal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
grd5: $t \in \mathbb{N}_1$
then
act1: $basal\_work := TRUE$
act2: $basal\_rate := r$
act3: $pump\_rate := r + normal\_rate + sd\_rate$
act4: $t\_basal := time + t$
end
Event basal_stop ⟨ordinary⟩ $\widehat{=}$
extends basal_stop
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
then
act1: $basal\_work := FALSE$
act2: $basal\_rate := 0$
act3: $pump\_rate := normal\_rate + sd\_rate$
act4: $t\_basal := 0$
end
Event basal_suspend ⟨ordinary⟩ $\widehat{=}$
extends basal_suspend
when
grd1: $basal\_work = TRUE$
grd3: $basal\_suspend = FALSE$

then

act1: $basal\_suspend := TRUE$

act2: $basal\_rate := 0$

act3: $pump\_rate := normal\_rate + sd\_rate$

act4: $t\_basal := 0$

end

Event basal_resume ⟨ordinary⟩ $\widehat{=}$

extends basal_resume

any

r

t

where

grd1: $basal\_suspend = TRUE$

grd2: $r \in \mathbb{N}$

grd3: $t \in \mathbb{N}_1$

then

act1: $basal\_suspend := FALSE$

act2: $basal\_rate := r$

act3: $pump\_rate := r + normal\_rate + sd\_rate$

act4: $t\_basal := t + time$

end

Event basal_update_rate ⟨ordinary⟩ $\widehat{=}$

extends basal_update_rate

any

r

t

where

grd2: $basal\_suspend = FALSE$

grd3: $basal\_work = TRUE$

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

grd5: $t\_basal = time$

grd6: $t \in \mathbb{N}_1$

then

act1: $basal\_rate := r$

act2: $pump\_rate := r + normal\_rate + sd\_rate$

act3: $t\_basal := time + t$

end

Event timer ⟨ordinary⟩ $\widehat{=}$

extends timer

when

grd1:

$\neg($

$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$

$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$

$(time = t\_sd)) \lor$

$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$

$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$

$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$

$)$

then

act1: $time := time + 1$

end

END

MACHINE control_Basal6
REFINES control6
SEES c_basal2
VARIABLES

normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
dmodule
d_update_time
basal_rate_in
basal_mode
btime
rate_setting2
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add
par_t_max
prog_basal
par_get_t
add_resume
add_update
add_start

INVARIANTS

Basal6.inv1: $prog\_basal \in PROC\_BASAL$
Basal6.inv3: $par\_get\_t \in 0 \mathinner{..} c-1$
Basal6.inv13: $add\_resume \in 0 \mathinner{..} 3$
Basal6.inv14: $add\_update \in 0 \mathinner{..} 3$
Basal6.inv15: $add\_start \in 0 \mathinner{..} 3$
inv1: $btime \in 1 \mathinner{..} c$
inv2: $par\_t \in \mathbb{N}$
inv3: $temp\_min \in 0 \mathinner{..} c$
inv4: $par\_t\_max \in 0 \mathinner{..} c-1$
inv5: $basal\_rate\_in \in 0 \mathinner{..} basal\_max$
inv6: $basal\_mode \in BASALMODE$
inv7: $rate\_setting2 \in 0 \mathinner{..} c-1 \rightarrow 0 \mathinner{..} basal\_max \cup \{-1\}$

inv8: $min\_value \in 0 \mathinner{..} c$

inv9: $max\_value \in 0 \mathinner{..} basal\_max$

inv10: $get\_min\_value\_add \in 0 \mathinner{..} 3$

inv11: $get\_max\_value\_add \in 0 \mathinner{..} 2$

inv12: $get\_min\_start\_t \in 0 \mathinner{..} c - 1$

inv13:
$get\_max\_start\_t \in 0 \mathinner{..} c - 1$

inv619: $rate\_setting2(0) \neq -1$

inv51: $get\_min\_value\_add \in 0 \mathinner{..} 3$

inv52: $par\_t \in \mathbb{N}$

inv53: $temp\_min \in 0 \mathinner{..} c$

inv54: $get\_min\_start\_t \in 0 \mathinner{..} c - 1$

inv57: $get\_min\_value\_add = 3 \Rightarrow \{i | i \in dom(rate\_setting2 \rhd \{-1\}) \wedge i > get\_min\_start\_t\} \neq \varnothing$

inv55: $get\_min\_value\_add = 3 \Rightarrow temp\_min = min(\{i | i \in dom(rate\_setting2 \rhd \{-1\}) \wedge i > get\_min\_start\_t\})$

inv58: $get\_min\_value\_add = 1 \Rightarrow par\_t = get\_min\_start\_t + 1$

inv510: $get\_min\_value\_add = 2 \Rightarrow par\_t > get\_min\_start\_t$

inv512: $get\_min\_value\_add = 2 \Rightarrow \{i | i \in dom(rate\_setting2 \rhd \{-1\}) \wedge i > get\_min\_start\_t \wedge i \leq par\_t - 1\} = \varnothing$

inv511: $get\_max\_start\_t \in 0 \mathinner{..} c - 1$

inv518: $get\_max\_value\_add \in 0 \mathinner{..} 2$

inv513: $par\_t\_max \in 0 \mathinner{..} c - 1$

inv514: $get\_max\_value\_add \in \{1, 2\} \Rightarrow get\_max\_start\_t \in 0 \mathinner{..} c - 1$

inv515: $get\_max\_value\_add = 2 \Rightarrow par\_t\_max = max(\{i | i \in dom(rate\_setting2 \rhd \{-1\}) \wedge i \leq get\_max\_start\_t\})$

inv516: $get\_max\_value\_add = 1 \Rightarrow \{i | i \in dom(rate\_setting2 \rhd \{-1\}) \wedge i \leq get\_max\_start\_t \wedge i \geq par\_t\_max + 1\} = \varnothing$

inv517:
$get\_max\_value\_add \in \{1, 2\} \Rightarrow par\_t\_max \leq get\_max\_start\_t$

inv61: $prog\_basal \in PROC\_BASAL$

inv62: $prog\_basal = null \Rightarrow get\_max\_value\_add = 0 \wedge get\_min\_value\_add = 0$

inv63: $par\_get\_t \in 0 \mathinner{..} c - 1$

inv64: $add\_resume \in 0 \mathinner{..} 3$

inv614: $add\_update \in 0 \mathinner{..} 3$

inv615: $add\_start \in 0 \mathinner{..} 3$

inv617: $prog\_basal = null \Rightarrow add\_resume = 0 \wedge add\_update = 0 \wedge add\_start = 0$

inv618: $add\_resume \neq 0 \Rightarrow add\_update = 0 \wedge add\_start = 0$

inv611: $get\_max\_value\_add \neq 0 \Rightarrow prog\_basal = call\_get\_max$

inv612: $get\_min\_value\_add \neq 0 \Rightarrow prog\_basal = call\_get\_min$

inv613: $prog\_basal \in \{call\_get\_min, return\_get\_min, call\_get\_max, return\_get\_max\} \Rightarrow par\_get\_t \in 0 \mathinner{..} c - 1$

inv68: $get\_min\_value\_add \in \{1, 2, 3\} \vee prog\_basal \in \{return\_get\_min, call\_get\_max, return\_get\_max\} \Rightarrow get\_min\_start\_t = par\_get\_t$

inv69: $get\_max\_value\_add \in \{1, 2\} \vee prog\_basal = return\_get\_max \Rightarrow get\_max\_start\_t = par\_get\_t$

inv65: $(add\_resume = 1 \wedge prog\_basal = return\_get\_min) \vee add\_resume = 2 \Rightarrow ((\forall j \cdot j \in dom(rate\_setting2) \wedge j > par\_get\_t \Rightarrow rate\_setting2(j) = -1) \Rightarrow min\_value = c)$

inv66: $(add\_resume = 1 \wedge prog\_basal = return\_get\_min) \vee add\_resume = 2 \Rightarrow ((\exists j \cdot j \in dom(rate\_setting2) \wedge j > par\_get\_t \wedge rate\_setting2(j) \neq -1) \Rightarrow min\_value = min(\{i | i \in dom(rate\_setting2 \rhd \{-1\}) \wedge i > par\_get\_t\}))$

inv67: $prog\_basal = return\_get\_max \Rightarrow max\_value = rate\_setting2(max(\{i | i \in dom(rate\_setting2 \rhd \{-1\}) \wedge i \leq get\_max\_start\_t\}))$

inv16: $add\_update \in \{1, 2\} \Rightarrow par\_get\_t \in dom(rate\_setting2 \rhd \{-1\})$

inv20: $(add\_start = 1 \wedge prog\_basal = return\_get\_min) \vee add\_start = 2 \Rightarrow ((\forall j \cdot j \in dom(rate\_setting2) \wedge j > par\_get\_t \Rightarrow rate\_setting2(j) = -1) \Rightarrow min\_value = c)$

inv19: $(add\_start = 1 \land prog\_basal = return\_get\_min) \lor add\_start = 2 \Rightarrow ((\exists j \cdot j \in dom(rate\_setting2) \land j > par\_get\_t \land rate\_setting2(j) \neq -1) \Rightarrow min\_value = min(\{i | i \in dom(rate\_setting2 \rhd \{-1\}) \land i > par\_get\_t\}))$

inv22: $add\_update = 1 \land prog\_basal = return\_get\_min \Rightarrow ((\forall j \cdot j \in dom(rate\_setting2) \land j > par\_get\_t \Rightarrow rate\_setting2(j) = -1) \Rightarrow min\_value = c)$

inv21: $add\_update = 1 \land prog\_basal = return\_get\_min \Rightarrow ((\exists j \cdot j \in dom(rate\_setting2) \land j > par\_get\_t \land rate\_setting2(j) \neq -1) \Rightarrow min\_value = min(\{i | i \in dom(rate\_setting2 \rhd \{-1\}) \land i > par\_get\_t\}))$

inv23: $add\_resume \in \{1, 2\} \Rightarrow basal\_rate\_in = 0 \land basal\_mode = suspended$

inv25: $add\_start \in \{1, 2\} \Rightarrow basal\_mode = stop$

inv26: $add\_start \neq 0 \Rightarrow add\_update = 0 \land add\_resume = 0$

inv627: $add\_update \neq 0 \Rightarrow add\_resume = 0 \land add\_start = 0$

inv624: $add\_update = 1 \Rightarrow basal\_mode = delivering \land prog\_basal \in \{call\_get\_min, return\_get\_min\}$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$
act2: $sd\_bolus\_work := FALSE$
act3: $sd\_preempted\_by\_normal := FALSE$
act7: $sd\_suspend := FALSE$
act8: $normal\_suspend := FALSE$
act9: $basal\_work := FALSE$
act10: $basal\_suspend := FALSE$
act11: $pump\_rate := 0$
act12: $basal\_rate := 0$
act13: $normal\_rate := 0$
act14: $sd\_rate := 0$
act15: $time := 0$
act16: $t\_basal := 0$
act17: $t\_normal := 0$
act18: $t\_sd := 0$
act19: $dmodule := FALSE$
act20: $d\_update\_time := 0$
Basal1.act4: $btime := c$
Basal1.act2: $basal\_rate\_in := 0$
Basal1.act3: $basal\_mode := stop$
Basal6.act15: $prog\_basal := null$
Basal6.act16: $par\_get\_t := 0$
Basal6.act17: $add\_resume := 0$
Basal6.act18: $add\_update := 0$
Basal6.act19: $add\_start := 0$
Basal6.act5: $rate\_setting2 := (1 \mathinner{..} c - 1 \times \{-1\}) \cup \{0 \mapsto 0\}$
Basal6.act6: $min\_value := 0$
Basal6.act7: $max\_value := 0$
Basal6.act11: $get\_min\_value\_add := 0$
Basal6.act8: $par\_t := 0$
Basal6.act9: $temp\_min := 0$
Basal6.act10: $get\_min\_start\_t := 0$
Basal6.act12: $get\_max\_start\_t := 0$
Basal6.act13: $get\_max\_value\_add := 0$
Basal6.act14: $par\_t\_max := 0$

end

Event control6.normal_bolus_start_1 ⟨ordinary⟩ $\widehat{=}$

extends normal_bolus_start_1

any

r

t
$t2$

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}$
grd6: $t \in \mathbb{N}$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \land time \neq d\_update\_time$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $normal\_bolus\_work := TRUE$
act2: $sd\_preempted\_by\_normal := TRUE$
act3: $normal\_rate := r$
act4: $sd\_rate := 0$
act5: $pump\_rate := r + basal\_rate$
act6: $t\_normal := time + t$
act7: $t\_sd := t\_sd - time$
act8: $d\_update\_time := t2$

end

Event control6.normal_bolus_start_2 ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_start_2

any

r
t

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
grd5: $t \in \mathbb{N}$

then

act1: $normal\_bolus\_work := TRUE$
act2: $normal\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_normal := time + t$

end

Event control6.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends normal_bolus_finish

when

grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
grd4: $time = t\_normal$

then

act1: $normal\_bolus\_work := FALSE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_normal := 0$

end

Event control6.normal_suspend ⟨ordinary⟩ $\widehat{=}$
extends normal_suspend

when

grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$

then

act1: $normal\_suspend := TRUE$
act2: $normal\_rate := 0$

act3: $pump\_rate := basal\_rate$
act4: $t\_normal := t\_normal - time$

end

Event control6.normal_resume ⟨ordinary⟩ $\widehat{=}$
extends normal_resume

when

grd1: $normal\_bolus\_work = TRUE$
grd2: $normal\_suspend = TRUE$
grd3: $\top$

then

act1: $normal\_bolus\_work := FALSE$
act2: $normal\_suspend := FALSE$
act4: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
act5: $t\_normal := 0$

end

Event control6.square_or_dual_bolus_start_s ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_start_s

any

r
t

where

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd4: $r \in \mathbb{N}_1$
grd5: $t \in \mathbb{N}_1$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$
act5: $dmodule := FALSE$

end

Event control6.square_or_dual_bolus_start_d ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_start_d

any

r
t
$t0$ t: both bolus, t0:for normal bolus

where

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd4: $r \in \mathbb{N}_1$
grd5: $t \in \mathbb{N}_1$
grd6: $t0 \in \mathbb{N}_1$
grd7: $t > t0$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$
act5: $dmodule := TRUE$
act6: $d\_update\_time := time + t0$

end

Event control6.square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_finish

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $dmodule := FALSE$

end

Event control6.square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_bolus_resume_from_normal

any

r
$t2$

where

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd7: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $d\_update\_time := t2$

end

Event control6.sd_suspend ⟨ordinary⟩ $\widehat{=}$
extends sd_suspend

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $sd\_suspend := TRUE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$
act6: $d\_update\_time := 0$

end

Event control6.square_or_dual_update_rate ⟨ordinary⟩ $\widehat{=}$
extends square_or_dual_update_rate

any

r

where

grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE$
grd7: $time = d\_update\_time$

then

act1: $sd\_rate := r$
act2: $pump\_rate := r + basal\_rate$
act3: $dmodule := FALSE$
act4: $d\_update\_time := 0$

end

Event control6.sd_resume ⟨ordinary⟩ $\widehat{=}$

extends sd_resume

when

grd1: $sd\_suspend = TRUE$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_suspend := FALSE$
act3: $sd\_rate := 0$
act4: $pump\_rate := basal\_rate$
act5: $t\_sd := 0$

end

Event control6.basal_start ⟨ordinary⟩ $\widehat{=}$

extends basal_start

any

r
t

where

grd1: $basal\_work = FALSE$
grd3: $basal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
grd5: $t \in \mathbb{N}_1$

then

act1: $basal\_work := TRUE$
act2: $basal\_rate := r$
act3: $pump\_rate := r + normal\_rate + sd\_rate$
act4: $t\_basal := time + t$

end

Event control6.basal_stop ⟨ordinary⟩ $\widehat{=}$

extends basal_stop

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

then

act1: $basal\_work := FALSE$
act2: $basal\_rate := 0$
act3: $pump\_rate := normal\_rate + sd\_rate$
act4: $t\_basal := 0$

end

Event control6.basal_suspend ⟨ordinary⟩ $\widehat{=}$

extends basal_suspend

when

grd1: $basal\_work = TRUE$
grd3: $basal\_suspend = FALSE$

then

act1: $basal\_suspend := TRUE$
act2: $basal\_rate := 0$
act3: $pump\_rate := normal\_rate + sd\_rate$
act4: $t\_basal := 0$

end

Event control6.basal_resume ⟨ordinary⟩ $\widehat{=}$

extends basal_resume

any

r
t

where

grd1: $basal\_suspend = TRUE$
grd2: $r \in \mathbb{N}$
grd3: $t \in \mathbb{N}_1$

then

act1: $basal\_suspend := FALSE$
act2: $basal\_rate := r$
act3: $pump\_rate := r + normal\_rate + sd\_rate$
act4: $t\_basal := t + time$

end

Event control6.basal_update_rate ⟨ordinary⟩ $\widehat{=}$

extends basal_update_rate

any

r
t

where

grd2: $basal\_suspend = FALSE$
grd3: $basal\_work = TRUE$
grd4: $r \in \mathbb{N}$
grd5: $t\_basal = time$
grd6: $t \in \mathbb{N}_1$

then

act1: $basal\_rate := r$
act2: $pump\_rate := r + normal\_rate + sd\_rate$
act3: $t\_basal := time + t$

end

Event control6.timer ⟨ordinary⟩ $\widehat{=}$

extends timer

when

grd1:
$\neg($
$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$
$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$
$(time = t\_sd)) \lor$
$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$
$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$
$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$
$)$

then

act1: $time := time + 1$

end

Event Basal6.basal_suspend ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd3: $prog\_basal = null$
Basal6.grd1: $basal\_rate\_in \neq 0$
Basal6.grd2: $basal\_mode = delivering$

then

Basal6.act1: $basal\_rate\_in := 0$
Basal6.act2: $basal\_mode := suspended$

end

Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$

any

t
r

where

Basal6.grd5: $prog\_basal = null$

Basal6.grd6: $t \in 0 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!\!- \{t \mapsto r\}$

end

Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$

any

t

where

Basal6.grd5: $prog\_basal = null$
Basal6.grd2: $basal\_mode \neq suspended$
Basal6.grd6: $t \in 1 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!\!- \{t \mapsto -1\}$

end

Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$

any

t
r

where

Basal6.grd9: $prog\_basal = null$
Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$
Basal6.grd4: $basal\_mode \neq suspended$
Basal6.grd5: $t \in 0 \mathinner{..} c - 1$
Basal6.grd6: $rate\_setting2(t) = -1$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!\!- \{t \mapsto r\}$

end

Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd8: $prog\_basal = return\_get\_max$
Basal6.grd9: $add\_resume = 2$

then

Basal6.act1: $basal\_rate\_in := max\_value$
Basal6.act2: $basal\_mode := delivering$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $prog\_basal := null$
Basal6.act5: $add\_resume := 0$

end

Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$

any

t

where

Basal6.grd4: $t \in 0 \mathinner{..} c - 1$
Basal6.grd5: $prog\_basal = null$
Basal6.grd6: $add\_resume = 0$
Basal6.grd1: $basal\_rate\_in = 0$
Basal6.grd3: $basal\_mode = suspended$

then

Basal6.act1: $par\_get\_t := t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_resume := 1$

end

Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd1: $prog\_basal = return\_get\_min$

Basal6.grd2: $add\_resume = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$

Basal6.act2: $add\_resume := 2$

end

Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd12: $add\_update = 1$

Basal6.grd4: $prog\_basal = return\_get\_min$

then

Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$

Basal6.act2: $btime := min\_value - par\_get\_t$

Basal6.act3: $add\_update := 0$

act4: $prog\_basal := null$

end

Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$

any

t

where

Basal6.grd6: $t \in 0 \mathinner{..} c - 1$

Basal6.grd2: $prog\_basal = null$

Basal6.grd3: $add\_update = 0$

Basal6.grd5: $basal\_mode = delivering$

Basal6.grd7: $rate\_setting2(t) \neq -1$

then

Basal6.act1: $par\_get\_t := t$

Basal6.act2: $prog\_basal := call\_get\_min$

Basal6.act3: $add\_update := 1$

end

Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd8: $add\_start = 2$

Basal6.grd9: $prog\_basal = return\_get\_max$

then

Basal6.act1: $basal\_mode := delivering$

Basal6.act2: $basal\_rate\_in := max\_value$

Basal6.act3: $btime := min\_value - par\_get\_t$

Basal6.act4: $add\_start := 0$

act4: $prog\_basal := null$

end

Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$

any

t

where

Basal6.grd1: $t \in 0 \mathinner{..} c - 1$

Basal6.grd2: $prog\_basal = null$

Basal6.grd3: $add\_start = 0$

Basal6.grd4: $basal\_mode = stop$

then

Basal6.act1: $par\_get\_t := t$

Basal6.act2: $prog\_basal := call\_get\_min$

Basal6.act3: $add\_start := 1$

end

Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd1: $prog\_basal = return\_get\_min$

Basal6.grd2: $add\_start = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_start := 2$

end

Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd2: $prog\_basal = null$
Basal6.grd1: $basal\_mode = delivering$

then

Basal6.act1: $basal\_mode := stop$
Basal6.act2: $basal\_rate\_in := 0$

end

Event Basal6.get_min_value_1 ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd4: $get\_min\_value\_add = 2$
Basal6.grd5: $par\_t = c$

then

Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$

end

Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd5: $get\_min\_value\_add = 3$

then

Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$

end

Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$

then

Basal6.act1: $par\_t := par\_get\_t + 1$
Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$

end

Event Basal6.find_min_value ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$

then

Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$

end

Event Basal6.find_min_value_2 ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$

then

Basal6.act1: $temp\_min := par\_t$
Basal6.act2: $get\_min\_value\_add := 3$

end

Event Basal6.get_max_value ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd2: $get\_max\_value\_add = 2$

then

Basal6.act3: $prog\_basal := return\_get\_max$

Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$

Basal6.act2: $get\_max\_value\_add := 0$

end

Event Basal6.get_max_value_start ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd2: $get\_max\_value\_add = 0$

Basal6.grd3: $prog\_basal = call\_get\_max$

then

Basal6.act1: $get\_max\_start\_t := par\_get\_t$

Basal6.act2: $get\_max\_value\_add := 1$

Basal6.act3: $par\_t\_max := par\_get\_t$

end

Event Basal6.get_max_value_1 ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd1: $get\_max\_value\_add = 1$

Basal6.grd3: $par\_t\_max \geq 0$

Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$

then

Basal6.act1: $par\_t\_max := par\_t\_max - 1$

end

Event Basal6.get_max_value_2 ⟨ordinary⟩ $\widehat{=}$

when

Basal6.grd1: $get\_max\_value\_add = 1$

Basal6.grd2: $par\_t\_max \geq 0$

Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$

then

Basal6.act1: $get\_max\_value\_add := 2$

end

END

MACHINE control_Basal6_2
REFINES control_Basal6
SEES c_prog
VARIABLES

normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
dmodule
d_update_time
basal_rate_in
basal_mode
btime
rate_setting2
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add
par_t_max
prog_basal
par_get_t
add_resume
add_update
add_start
prog
par_basal_start_t
par_basal_resume_t
par_basal_update_rate_t

INVARIANTS

inv1: $prog \in PROG$
inv2: $prog = call\_basal\_start \Rightarrow par\_basal\_start\_t \in 0 \mathinner{..} c - 1$
inv8: $par\_basal\_resume\_t \in \mathbb{N}$
inv9: $prog = call\_basal\_resume \Rightarrow par\_basal\_resume\_t \in 0 \mathinner{..} c - 1$
inv5: $prog = return\_basal\_start \Rightarrow basal\_rate\_in \in \mathbb{N}$
inv6: $prog = return\_basal\_start \Rightarrow btime \in \mathbb{N}_1$
inv12: $par\_basal\_update\_rate\_t \in \mathbb{N}$
inv13: $prog = call\_basal\_update \Rightarrow par\_basal\_update\_rate\_t \in dom(rate\_setting2 \rhd \{-1\})$

inv14: $prog = return\_basal\_suspend \Rightarrow basal\_rate\_in = 0$

inv15: $prog = return\_basal\_stop \Rightarrow basal\_rate\_in = 0$

inv3: $prog = call\_basal\_suspend \Rightarrow basal\_rate\_in \neq 0$

inv42: $add\_resume \in \{1, 2\} \Rightarrow prog = call\_basal\_resume$

inv43: $add\_update = 1 \Rightarrow prog = call\_basal\_update$

inv44: $add\_start \in \{1, 2\} \Rightarrow prog = call\_basal\_start$

inv27: $prog = null \Rightarrow (basal\_work = TRUE \wedge basal\_suspend = FALSE \Rightarrow basal\_mode = delivering)$

inv26: $prog = null \Rightarrow (basal\_rate \neq 0 \Leftrightarrow basal\_rate\_in \neq 0)$

inv25: $prog = return\_basal\_update \Rightarrow (basal\_work = TRUE \wedge basal\_suspend = FALSE \Rightarrow basal\_mode = delivering)$

inv24: $prog = return\_basal\_resume \Rightarrow basal\_mode = delivering$

inv23: $prog = return\_basal\_start \Rightarrow basal\_mode = delivering$

inv22: $prog = call\_basal\_suspend \Rightarrow basal\_rate\_in \neq 0 \wedge basal\_mode = delivering$

inv21: $prog = call\_basal\_resume \Rightarrow basal\_rate\_in = 0 \wedge basal\_mode = suspended$

inv20: $prog = call\_basal\_update \Rightarrow basal\_mode = delivering$

inv19: $prog = call\_basal\_start \Rightarrow basal\_mode = stop$

inv18: $prog = call\_basal\_stop \Rightarrow basal\_mode = delivering$

inv17: $prog = null \wedge basal\_work = FALSE \wedge basal\_suspend = FALSE \Rightarrow basal\_mode = stop$

inv16: $prog = return\_basal\_stop \Rightarrow basal\_mode = stop$

inv29: $basal\_suspend = TRUE \wedge prog = null \wedge basal\_work = TRUE \Rightarrow basal\_mode = suspended$

inv41: $prog = return\_basal\_suspend \Rightarrow basal\_mode = suspended$

inv30: $prog = return\_basal\_start \Rightarrow basal\_work = FALSE \wedge basal\_suspend = FALSE$

inv31: $prog = call\_basal\_start \Rightarrow basal\_work = FALSE \wedge basal\_suspend = FALSE$

inv32: $prog = return\_basal\_stop \Rightarrow basal\_suspend = FALSE \wedge basal\_work = TRUE$

inv33: $prog = call\_basal\_stop \Rightarrow basal\_suspend = FALSE \wedge basal\_work = TRUE$

inv34: $prog = return\_basal\_suspend \Rightarrow basal\_suspend = FALSE \wedge basal\_work = TRUE$

inv35: $prog = call\_basal\_suspend \Rightarrow basal\_suspend = FALSE \wedge basal\_work = TRUE$

inv36: $prog = return\_basal\_resume \Rightarrow basal\_suspend = TRUE$

inv37: $prog = call\_basal\_resume \Rightarrow basal\_suspend = TRUE$

inv38: $prog = return\_basal\_update \Rightarrow basal\_suspend = FALSE \wedge basal\_work = TRUE \wedge t\_basal = time$

inv39: $prog = call\_basal\_update \Rightarrow basal\_suspend = FALSE \wedge basal\_work = TRUE \wedge t\_basal = time$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$

act2: $sd\_bolus\_work := FALSE$

act3: $sd\_preempted\_by\_normal := FALSE$

act7: $sd\_suspend := FALSE$

act8: $normal\_suspend := FALSE$

act9: $basal\_work := FALSE$

act10: $basal\_suspend := FALSE$

act11: $pump\_rate := 0$

act12: $basal\_rate := 0$

act13: $normal\_rate := 0$

act14: $sd\_rate := 0$

act15: $time := 0$

act16: $t\_basal := 0$

act17: $t\_normal := 0$

act18: $t\_sd := 0$
act19: $dmodule := FALSE$
act20: $d\_update\_time := 0$
Basal1.act4: $btime := c$
Basal1.act2: $basal\_rate\_in := 0$
Basal1.act3: $basal\_mode := stop$
Basal6.act15: $prog\_basal := null$
Basal6.act16: $par\_get\_t := 0$
Basal6.act17: $add\_resume := 0$
Basal6.act18: $add\_update := 0$
Basal6.act19: $add\_start := 0$
Basal6.act5: $rate\_setting2 := (1 \mathinner{..} c-1 \times \{-1\}) \cup \{0 \mapsto 0\}$
Basal6.act6: $min\_value := 0$
Basal6.act7: $max\_value := 0$
Basal6.act11: $get\_min\_value\_add := 0$
Basal6.act8: $par\_t := 0$
Basal6.act9: $temp\_min := 0$
Basal6.act10: $get\_min\_start\_t := 0$
Basal6.act12: $get\_max\_start\_t := 0$
Basal6.act13: $get\_max\_value\_add := 0$
Basal6.act14: $par\_t\_max := 0$
act21: $prog := null$
act22: $par\_basal\_start\_t := 0$
act23: $par\_basal\_resume\_t := 0$
act24: $par\_basal\_update\_rate\_t := 0$

end

Event control6.normal_bolus_start_1 ⟨ordinary⟩ $\widehat{=}$
extends control6.normal_bolus_start_1

any

r
t
$t2$

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}$
grd6: $t \in \mathbb{N}$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \wedge time \neq d\_update\_time$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $normal\_bolus\_work := TRUE$
act2: $sd\_preempted\_by\_normal := TRUE$
act3: $normal\_rate := r$
act4: $sd\_rate := 0$
act5: $pump\_rate := r + basal\_rate$
act6: $t\_normal := time + t$
act7: $t\_sd := t\_sd - time$
act8: $d\_update\_time := t2$

end

Event control6.normal_bolus_start_2 ⟨ordinary⟩ $\widehat{=}$
extends control6.normal_bolus_start_2

any

r
t

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = FALSE$

grd3: $normal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
grd5: $t \in \mathbb{N}$

then

act1: $normal\_bolus\_work := TRUE$
act2: $normal\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_normal := time + t$

end

Event control6.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends control6.normal_bolus_finish

when

grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
grd4: $time = t\_normal$

then

act1: $normal\_bolus\_work := FALSE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_normal := 0$

end

Event control6.normal_suspend ⟨ordinary⟩ $\widehat{=}$
extends control6.normal_suspend

when

grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$

then

act1: $normal\_suspend := TRUE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_normal := t\_normal - time$

end

Event control6.normal_resume ⟨ordinary⟩ $\widehat{=}$
extends control6.normal_resume

when

grd1: $normal\_bolus\_work = TRUE$
grd2: $normal\_suspend = TRUE$
grd3: $\top$

then

act1: $normal\_bolus\_work := FALSE$
act2: $normal\_suspend := FALSE$
act4: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
act5: $t\_normal := 0$

end

Event control6.square_or_dual_bolus_start_s ⟨ordinary⟩ $\widehat{=}$
extends control6.square_or_dual_bolus_start_s

any

r
t

where

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd4: $r \in \mathbb{N}_1$
grd5: $t \in \mathbb{N}_1$

then

act1: $sd\_bolus\_work := TRUE$

act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$
act5: $dmodule := FALSE$

end

Event control6.square_or_dual_bolus_start_d ⟨ordinary⟩ $\widehat{=}$

extends control6.square_or_dual_bolus_start_d

any

r
t
$t0$ t: both bolus, t0:for normal bolus

where

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd4: $r \in \mathbb{N}_1$
grd5: $t \in \mathbb{N}_1$
grd6: $t0 \in \mathbb{N}_1$
grd7: $t > t0$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$
act5: $dmodule := TRUE$
act6: $d\_update\_time := time + t0$

end

Event control6.square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$

extends control6.square_or_dual_bolus_finish

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $dmodule := FALSE$

end

Event control6.square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$

extends control6.square_or_dual_bolus_resume_from_normal

any

r
$t2$

where

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd7: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$

act4: $t\_sd := time + t\_sd$
act5: $d\_update\_time := t2$

end

Event control6.sd_suspend ⟨ordinary⟩ $\widehat{=}$

extends control6.sd_suspend

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $sd\_suspend := TRUE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$
act6: $d\_update\_time := 0$

end

Event control6.square_or_dual_update_rate ⟨ordinary⟩ $\widehat{=}$

extends control6.square_or_dual_update_rate

any

r

where

grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE$
grd7: $time = d\_update\_time$

then

act1: $sd\_rate := r$
act2: $pump\_rate := r + basal\_rate$
act3: $dmodule := FALSE$
act4: $d\_update\_time := 0$

end

Event control6.sd_resume ⟨ordinary⟩ $\widehat{=}$

extends control6.sd_resume

when

grd1: $sd\_suspend = TRUE$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_suspend := FALSE$
act3: $sd\_rate := 0$
act4: $pump\_rate := basal\_rate$
act5: $t\_sd := 0$

end

Event control5·basal_suspend_return ⟨ordinary⟩ $\widehat{=}$

refines control6.basal_suspend

when

grd1: $prog = return\_basal\_suspend$

then

control5.act4: $t\_basal := 0$
control5.act1: $basal\_suspend := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$

end

Event control5·basal_suspend_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

grd4: $basal\_rate \neq 0$
grd3: $prog = null$

then

act1: $prog := call\_basal\_suspend$

end

Event control5·basal_resume_return ⟨ordinary⟩ $\widehat{=}$

refines control6.basal_resume

when

grd1: $prog = return\_basal\_resume$

with

r: $r = basal\_rate\_in$
t: $t = btime$

then

control5.act1: $basal\_suspend := FALSE$
control5.act4: $t\_basal := btime + time$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$

end

Event control5·basal_resume_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $basal\_suspend = TRUE$
grd2: $prog = null$

then

act1: $prog := call\_basal\_resume$
act2: $par\_basal\_resume\_t := timemodc$

end

Event control5·basal_update_rate_return ⟨ordinary⟩ $\widehat{=}$

refines control6.basal_update_rate

when

grd1: $prog = return\_basal\_update$

with

t: $t = btime$
r: $r = basal\_rate\_in$

then

control5.act3: $t\_basal := time + btime$
control5.act1: $basal\_rate := basal\_rate\_in$
control5.act2: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$

end

Event control5·basal_update_rate_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $t\_basal = time$
grd2: $basal\_suspend = FALSE$
grd3: $basal\_work = TRUE$
grd4: $prog = null$
grd5: $t\_basal \in dom(rate\_setting2 \rhd \{-1\})$

then

act1: $prog := call\_basal\_update$
act2: $par\_basal\_update\_rate\_t := t\_basal$

end

Event control5·basal_start_return ⟨ordinary⟩ $\widehat{=}$

refines control6.basal_start

when

grd1: $prog = return\_basal\_start$

with
t: $t = btime$
r: $r = basal\_rate\_in$
then
control5.act4: $t\_basal := time + btime$
control5.act1: $basal\_work := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
end

Event control5·basal_start_call ⟨ordinary⟩ $\widehat{=}$
when
grd1: $basal\_work = FALSE$
grd2: $basal\_suspend = FALSE$
grd3: $prog = null$
then
act1: $prog := call\_basal\_start$
act2: $par\_basal\_start\_t := timemodc$
end

Event control5·basal_stop_return ⟨ordinary⟩ $\widehat{=}$
refines control6.basal_stop
when
grd1: $prog = return\_basal\_stop$
then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_work := FALSE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
end

Event control5·basal_stop_call ⟨ordinary⟩ $\widehat{=}$
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd3: $prog = null$
then
act1: $prog := call\_basal\_stop$
end

Event control6.timer ⟨ordinary⟩ $\widehat{=}$
extends control6.timer
when
grd1:
$\neg($
$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$
$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$
$(time = t\_sd)) \lor$
$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$
$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$
$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$
$)$
then
act1: $time := time + 1$
end

Event Basal6.basal_suspend ⟨ordinary⟩ $\widehat{=}$
refines Basal6.basal_suspend
when
grd1: $prog = call\_basal\_suspend$
Basal6.grd3: $prog\_basal = null$

Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in \neq 0$
Basal6.grd2: ⟨theorem⟩ $basal\_mode = delivering$

then

Basal6.act1: $basal\_rate\_in := 0$
Basal6.act2: $basal\_mode := suspended$
act1: $prog := return\_basal\_suspend$

end

Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6·change_setting

any

t
r

where

Basal6.grd5: $prog\_basal = null$
Basal6.grd6: $t \in 0 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$

end

Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.delete_setting

any

t

where

Basal6.grd5: $prog\_basal = null$
Basal6.grd2: $basal\_mode \neq suspended$
Basal6.grd6: $t \in 1 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
grd1: $t \neq par\_basal\_update\_rate\_t$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto -1\}$

end

Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.add_setting

any

t
r

where

Basal6.grd9: $prog\_basal = null$
Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$
Basal6.grd4: $basal\_mode \neq suspended$
Basal6.grd5: $t \in 0 \mathinner{..} c - 1$
Basal6.grd6: $rate\_setting2(t) = -1$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$

end

Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_return

when

Basal6.grd8: $prog\_basal = return\_get\_max$
Basal6.grd9: $add\_resume = 2$

then

Basal6.act1: $basal\_rate\_in := max\_value$
Basal6.act2: $basal\_mode := delivering$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $prog\_basal := null$
Basal6.act5: $add\_resume := 0$

act1: $prog := return\_basal\_resume$

end

Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$

refines Basal6.basal_resume_call

when

grd1: $prog = call\_basal\_resume$

Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in = 0$

Basal6.grd3: ⟨theorem⟩ $basal\_mode = suspended$

Basal6.grd6: $add\_resume = 0$

Basal6.grd5: $prog\_basal = null$

with

t: $t = par\_basal\_resume\_t$

then

Basal6.act1: $par\_get\_t := par\_basal\_resume\_t$

Basal6.act2: $prog\_basal := call\_get\_min$

Basal6.act3: $add\_resume := 1$

end

Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$

Basal6.grd2: $add\_resume = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$

Basal6.act2: $add\_resume := 2$

end

Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6·rate_update_return

when

Basal6.grd12: $add\_update = 1$

Basal6.grd4: $prog\_basal = return\_get\_min$

then

Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$

Basal6.act2: $btime := min\_value - par\_get\_t$

Basal6.act3: $add\_update := 0$

act4: $prog\_basal := null$

act5: $prog := return\_basal\_update$

end

Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$

refines Basal6.rate_update_call

when

grd1: $prog = call\_basal\_update$

Basal6.grd5: ⟨theorem⟩ $basal\_mode = delivering$

Basal6.grd7: ⟨theorem⟩ $rate\_setting2(par\_basal\_update\_rate\_t) \neq -1$

Basal6.grd3: $add\_update = 0$

Basal6.grd2: $prog\_basal = null$

with

t: $t = par\_basal\_update\_rate\_t$

then

Basal6.act1: $par\_get\_t := par\_basal\_update\_rate\_t$

Basal6.act2: $prog\_basal := call\_get\_min$

Basal6.act3: $add\_update := 1$

end

Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_return

when

Basal6.grd8: $add\_start = 2$

Basal6.grd9: $prog\_basal = return\_get\_max$

then

Basal6.act1: $basal\_mode := delivering$
Basal6.act2: $basal\_rate\_in := max\_value$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $add\_start := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_start$

end

Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$

refines Basal6.start_call

when

grd1: $prog = call\_basal\_start$
Basal6.grd3: $add\_start = 0$
Basal6.grd4: ⟨theorem⟩ $basal\_mode = stop$
Basal6.grd2: $prog\_basal = null$

with

t: $t = par\_basal\_start\_t$

then

Basal6.act1: $par\_get\_t := par\_basal\_start\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_start := 1$

end

Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_start = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_start := 2$

end

Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$

extends Basal6.stop

when

Basal6.grd2: $prog\_basal = null$
Basal6.grd1: $basal\_mode = delivering$
grd1: $prog = call\_basal\_stop$

then

Basal6.act1: $basal\_mode := stop$
Basal6.act2: $basal\_rate\_in := 0$
act1: $prog := return\_basal\_stop$

end

Event Basal6.get_min_value_1 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_min_value_1

when

Basal6.grd4: $get\_min\_value\_add = 2$
Basal6.grd5: $par\_t = c$

then

Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$

end

Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_min_value_2

when

Basal6.grd5: $get\_min\_value\_add = 3$

then
Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_start
when
Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$
then
Basal6.act1: $par\_t := par\_get\_t + 1$
Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$
end
Event Basal6.find_min_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$
then
Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$
end
Event Basal6.find_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value_2
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$
then
Basal6.act1: $temp\_min := par\_t$
Basal6.act2: $get\_min\_value\_add := 3$
end
Event Basal6.get_max_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value
when
Basal6.grd2: $get\_max\_value\_add = 2$
then
Basal6.act3: $prog\_basal := return\_get\_max$
Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$
Basal6.act2: $get\_max\_value\_add := 0$
end
Event Basal6.get_max_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_start
when
Basal6.grd2: $get\_max\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_max$
then
Basal6.act1: $get\_max\_start\_t := par\_get\_t$
Basal6.act2: $get\_max\_value\_add := 1$
Basal6.act3: $par\_t\_max := par\_get\_t$
end
Event Basal6.get_max_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_1
when

Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd3: $par\_t\_max \geq 0$
Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$

then

Basal6.act1: $par\_t\_max := par\_t\_max - 1$

end

Event Basal6.get_max_value_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_max_value_2

when

Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd2: $par\_t\_max \geq 0$
Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$

then

Basal6.act1: $get\_max\_value\_add := 2$

end

END

MACHINE control_Basal6_NormalBolus
REFINES control_Basal6_2
SEES c_normalbolus,c_prog
VARIABLES

rate_setting2
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
basal_rate_in
basal_mode
btime
prog
par_basal_start_t
par_basal_resume_t
par_basal_update_rate_t
insulin_needed
normal_add
normal_delivering_time
normal_delivering_rate
normal_bolus_suspend
dmodule
d_update_time
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add
par_t_max
prog_basal
par_get_t
add_resume
add_update
add_start

INVARIANTS

NormalBolus.inv1: $insulin\_needed \in \mathbb{N}$
NormalBolus.inv5: $normal\_add \in 0 \mathinner{..} 3$
NormalBolus.inv2: $normal\_delivering\_time \in \mathbb{N}$

NormalBolus.inv3: $normal\_delivering\_rate \in \mathbb{N}$

NormalBolus.inv4: $normal\_delivering\_rate = 0 \lor normal\_delivering\_rate = normal\_bolus\_rate$

NormalBolus.inv6: $normal\_add = 0 \Rightarrow normal\_delivering\_rate = 0$

NormalBolus.inv7: $normal\_add = 1 \Rightarrow insulin\_needed \neq 0 \land normal\_delivering\_rate = 0$

NormalBolus.inv9: $normal\_add = 2 \Rightarrow normal\_delivering\_rate = 0$

NormalBolus.inv8: $normal\_add = 3 \Rightarrow normal\_delivering\_rate = normal\_bolus\_rate$

NormalBolus.inv10: $normal\_bolus\_suspend \in BOOL$

NormalBolus.inv11: $normal\_add = 1 \Rightarrow normal\_bolus\_suspend = FALSE$

NormalBolus.inv12: $normal\_add = 2 \Rightarrow normal\_bolus\_suspend = FALSE$

inv19: $normal\_add = 1 \Rightarrow normal\_bolus\_suspend = FALSE \land normal\_delivering\_rate = 0 \land normal\_delivering\_time = 0$

inv20: $normal\_add = 2 \Rightarrow normal\_bolus\_suspend = FALSE \land normal\_delivering\_rate = 0$

inv21: $normal\_add = 3 \Rightarrow normal\_bolus\_suspend = FALSE \land normal\_delivering\_rate > 0$

inv22: $normal\_add = 0 \Rightarrow normal\_delivering\_rate = 0 \land normal\_delivering\_time = 0$

inv23: ⟨theorem⟩ $normal\_bolus\_suspend = TRUE \Rightarrow normal\_add = 0$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$

act2: $sd\_bolus\_work := FALSE$

act3: $sd\_preempted\_by\_normal := FALSE$

act7: $sd\_suspend := FALSE$

act8: $normal\_suspend := FALSE$

act9: $basal\_work := FALSE$

act10: $basal\_suspend := FALSE$

act11: $pump\_rate := 0$

act12: $basal\_rate := 0$

act13: $normal\_rate := 0$

act14: $sd\_rate := 0$

act15: $time := 0$

act16: $t\_basal := 0$

act17: $t\_normal := 0$

act18: $t\_sd := 0$

act19: $dmodule := FALSE$

act20: $d\_update\_time := 0$

Basal1.act4: $btime := c$

Basal1.act2: $basal\_rate\_in := 0$

Basal1.act3: $basal\_mode := stop$

Basal6.act15: $prog\_basal := null$

Basal6.act16: $par\_get\_t := 0$

Basal6.act17: $add\_resume := 0$

Basal6.act18: $add\_update := 0$

Basal6.act19: $add\_start := 0$

Basal6.act5: $rate\_setting2 := (1 \mathinner{..} c - 1 \times \{-1\}) \cup \{0 \mapsto 0\}$

Basal6.act6: $min\_value := 0$

Basal6.act7: $max\_value := 0$

Basal6.act11: $get\_min\_value\_add := 0$

Basal6.act8: $par\_t := 0$

Basal6.act9: $temp\_min := 0$

Basal6.act10: $get\_min\_start\_t := 0$

Basal6.act12: $get\_max\_start\_t := 0$

Basal6.act13: $get\_max\_value\_add := 0$

Basal6.act14: $par\_t\_max := 0$

act21: $prog := null$

act22: $par\_basal\_start\_t := 0$

act23: $par\_basal\_resume\_t := 0$

act24: $par\_basal\_update\_rate\_t := 0$

NormalBolus.act1: $insulin\_needed := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_delivering\_rate := 0$
NormalBolus.act4: $normal\_add := 0$
NormalBolus.act5: $normal\_bolus\_suspend := FALSE$

end

Event control5.normal_bolus_start_1 ⟨ordinary⟩ $\widehat{=}$

extends control6.normal_bolus_start_1

any

r
t
$t2$

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}$
grd6: $t \in \mathbb{N}$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \land time \neq d\_update\_time$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $normal\_bolus\_work := TRUE$
act2: $sd\_preempted\_by\_normal := TRUE$
act3: $normal\_rate := r$
act4: $sd\_rate := 0$
act5: $pump\_rate := r + basal\_rate$
act6: $t\_normal := time + t$
act7: $t\_sd := t\_sd - time$
act8: $d\_update\_time := t2$

end

Event control5.normal_bolus_start_2 ⟨ordinary⟩ $\widehat{=}$

extends control6.normal_bolus_start_2

any

r
t

where

grd1: $normal\_bolus\_work = FALSE$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_suspend = FALSE$
grd4: $r \in \mathbb{N}$
grd5: $t \in \mathbb{N}$

then

act1: $normal\_bolus\_work := TRUE$
act2: $normal\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_normal := time + t$

end

Event control5.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$

extends control6.normal_bolus_finish

when

grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
grd4: $time = t\_normal$

then

act1: $normal\_bolus\_work := FALSE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$

act4: $t\_normal := 0$

end

Event control5·square_or_dual_bolus_start_s ⟨ordinary⟩ $\widehat{=}$

extends control6.square_or_dual_bolus_start_s

any

r

t

where

grd1: $sd\_bolus\_work = FALSE$

grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_suspend = FALSE$

grd4: $r \in \mathbb{N}_1$

grd5: $t \in \mathbb{N}_1$

then

act1: $sd\_bolus\_work := TRUE$

act2: $sd\_rate := r$

act3: $pump\_rate := r + basal\_rate$

act4: $t\_sd := time + t$

act5: $dmodule := FALSE$

end

Event control5·square_or_dual_bolus_start_d ⟨ordinary⟩ $\widehat{=}$

extends control6.square_or_dual_bolus_start_d

any

r

t

$t0$ t: both bolus, t0:for normal bolus

where

grd1: $sd\_bolus\_work = FALSE$

grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_suspend = FALSE$

grd4: $r \in \mathbb{N}_1$

grd5: $t \in \mathbb{N}_1$

grd6: $t0 \in \mathbb{N}_1$

grd7: $t > t0$

then

act1: $sd\_bolus\_work := TRUE$

act2: $sd\_rate := r$

act3: $pump\_rate := r + basal\_rate$

act4: $t\_sd := time + t$

act5: $dmodule := TRUE$

act6: $d\_update\_time := time + t0$

end

Event control5.square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$

extends control6.square_or_dual_bolus_finish

when

grd1: $sd\_bolus\_work = TRUE$

grd2: $sd\_preempted\_by\_normal = FALSE$

grd4: $sd\_suspend = FALSE$

grd5: $time = t\_sd$

then

act1: $sd\_bolus\_work := FALSE$

act2: $sd\_rate := 0$

act3: $pump\_rate := basal\_rate$

act4: $t\_sd := 0$

act5: $dmodule := FALSE$

end

Event control5.square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$

extends control6.square_or_dual_bolus_resume_from_normal

any
r
$t2$
where
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd7: $dmodule = FALSE \Rightarrow t2 = 0$
then
act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $d\_update\_time := t2$
end

Event control5.normal_suspend ⟨ordinary⟩ $\widehat{=}$
extends control6.normal_suspend
when
grd1: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
then
act1: $normal\_suspend := TRUE$
act2: $normal\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_normal := t\_normal - time$
end

Event control5.sd_suspend ⟨ordinary⟩ $\widehat{=}$
extends control6.sd_suspend
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $sd\_suspend := TRUE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$
act6: $d\_update\_time := 0$
end

Event control5.square_or_dual_update_rate ⟨ordinary⟩ $\widehat{=}$
extends control6.square_or_dual_update_rate
any
r
where
grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE$
grd7: $time = d\_update\_time$
then
act1: $sd\_rate := r$
act2: $pump\_rate := r + basal\_rate$
act3: $dmodule := FALSE$

act4: $d\_update\_time := 0$

end

Event control5.normal_resume ⟨ordinary⟩ $\widehat{=}$

extends control6.normal_resume

when

grd1: $normal\_bolus\_work = TRUE$

grd2: $normal\_suspend = TRUE$

grd3: $\top$

then

act1: $normal\_bolus\_work := FALSE$

act2: $normal\_suspend := FALSE$

act4: $normal\_rate := 0$

act3: $pump\_rate := basal\_rate$

act5: $t\_normal := 0$

end

Event control5.sd_resume ⟨ordinary⟩ $\widehat{=}$

extends control6.sd_resume

when

grd1: $sd\_suspend = TRUE$

then

act1: $sd\_bolus\_work := FALSE$

act2: $sd\_suspend := FALSE$

act3: $sd\_rate := 0$

act4: $pump\_rate := basal\_rate$

act5: $t\_sd := 0$

end

Event control5·basal_start_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_return

when

grd1: $prog = return\_basal\_start$

then

control5.act4: $t\_basal := time + btime$

control5.act1: $basal\_work := TRUE$

control5.act2: $basal\_rate := basal\_rate\_in$

control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$

act1: $prog := null$

end

Event control5·basal_start_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_call

when

grd1: $basal\_work = FALSE$

grd2: $basal\_suspend = FALSE$

grd3: $prog = null$

then

act1: $prog := call\_basal\_start$

act2: $par\_basal\_start\_t := timemodc$

end

Event control5·basal_stop_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_stop_return

when

grd1: $prog = return\_basal\_stop$

then

control5.act4: $t\_basal := 0$

control5.act1: $basal\_work := FALSE$

control5.act2: $basal\_rate := basal\_rate\_in$

control5.act3: $pump\_rate := normal\_rate + sd\_rate$

act1: $prog := null$

end
Event control5·basal_stop_call ⟨ordinary⟩ ≙
extends control5·basal_stop_call
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd3: $prog = null$
then
act1: $prog := call\_basal\_stop$
end
Event control5·basal_suspend_return ⟨ordinary⟩ ≙
extends control5·basal_suspend_return
when
grd1: $prog = return\_basal\_suspend$
then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_suspend := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
end
Event control5·basal_suspend_call ⟨ordinary⟩ ≙
extends control5·basal_suspend_call
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

grd4: $basal\_rate \neq 0$
grd3: $prog = null$
then
act1: $prog := call\_basal\_suspend$
end
Event control5·basal_resume_return ⟨ordinary⟩ ≙
extends control5·basal_resume_return
when
grd1: $prog = return\_basal\_resume$
then
control5.act1: $basal\_suspend := FALSE$
control5.act4: $t\_basal := btime + time$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
end
Event control5·basal_resume_call ⟨ordinary⟩ ≙
extends control5·basal_resume_call
when
grd1: $basal\_suspend = TRUE$
grd2: $prog = null$
then
act1: $prog := call\_basal\_resume$
act2: $par\_basal\_resume\_t := timemodc$
end
Event control5·basal_update_rate_return ⟨ordinary⟩ ≙
extends control5 ·basal_update_rate_return
when
grd1: $prog = return\_basal\_update$
then

control5.act3: $t\_basal := time + btime$
control5.act1: $basal\_rate := basal\_rate\_in$
control5.act2: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$

end

Event control5·basal_update_rate_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_update_rate_call

when

grd1: $t\_basal = time$
grd2: $basal\_suspend = FALSE$
grd3: $basal\_work = TRUE$
grd4: $prog = null$
grd5: $t\_basal \in dom(rate\_setting2 \rhd \{-1\})$

then

act1: $prog := call\_basal\_update$
act2: $par\_basal\_update\_rate\_t := t\_basal$

end

Event control5.timer ⟨ordinary⟩ $\widehat{=}$

extends control6.timer

when

grd1:
$\neg($
$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$
$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$
$(time = t\_sd)) \lor$
$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$
$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$
$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$
$)$

then

act1: $time := time + 1$

end

Event NormalBolus.normal_bolus_start_calculate_insulin_needed ⟨ordinary⟩ $\widehat{=}$

any

insulin

where

NormalBolus.grd1: $insulin > 0$
NormalBolus.grd3: $normal\_add = 0$
NormalBolus.grd4: $normal\_bolus\_suspend = FALSE$

then

NormalBolus.act1: $insulin\_needed := insulin$
NormalBolus.act2: $normal\_add := 1$

end

Event NormalBolus.normal_bolus_start_calculate_lasting_time ⟨ordinary⟩ $\widehat{=}$

when

NormalBolus.grd1: $normal\_add = 1$

then

NormalBolus.act1: $normal\_delivering\_time := insulin\_needed / normal\_bolus\_rate$
NormalBolus.act2: $insulin\_needed := 0$
NormalBolus.act3: $normal\_add := 2$

end

Event NormalBolus.normal_bolus_delivery ⟨ordinary⟩ $\widehat{=}$

when

NormalBolus.grd2: $normal\_add = 2$

then

NormalBolus.act1: $normal\_delivering\_rate := normal\_bolus\_rate$
NormalBolus.act2: $normal\_add := 3$

end

Event NormalBolus.normal_bolus_suspend ⟨ordinary⟩ $\widehat{=}$
when
NormalBolus.grd4: $normal\_add = 3$
NormalBolus.grd5: $normal\_bolus\_suspend = FALSE$
then
NormalBolus.act1: $normal\_delivering\_rate := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_add := 0$
NormalBolus.act4: $normal\_bolus\_suspend := TRUE$
end
Event NormalBolus.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$
when
NormalBolus.grd4: $normal\_bolus\_suspend = FALSE$
NormalBolus·grd3: $normal\_add = 3$
then
NormalBolus.act1: $normal\_delivering\_rate := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_add := 0$
end
Event NormalBolus·normal_bolus_resume ⟨ordinary⟩ $\widehat{=}$
when
NormalBolus.grd1: $normal\_bolus\_suspend = TRUE$
NormalBolus.grd2: $normal\_add = 0$
then
NormalBolus.act1: $normal\_bolus\_suspend := FALSE$
act1: $normal\_delivering\_rate := 0$
end
Event Basal6.basal_suspend ⟨ordinary⟩ $\widehat{=}$
extends Basal6.basal_suspend
when
grd1: $prog = call\_basal\_suspend$
Basal6.grd3: $prog\_basal = null$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in \neq 0$
Basal6.grd2: ⟨theorem⟩ $basal\_mode = delivering$
then
Basal6.act1: $basal\_rate\_in := 0$
Basal6.act2: $basal\_mode := suspended$
act1: $prog := return\_basal\_suspend$
end
Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6·change_setting
any
t
r
where
Basal6.grd5: $prog\_basal = null$
Basal6.grd6: $t \in 0 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$
then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$
end
Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6.delete_setting
any
t
where
Basal6.grd5: $prog\_basal = null$

Basal6.grd2: $basal\_mode \neq suspended$
Basal6.grd6: $t \in 1 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
grd1: $t \neq par\_basal\_update\_rate\_t$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto -1\}$

end

Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.add_setting

any

t
r

where

Basal6.grd9: $prog\_basal = null$
Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$
Basal6.grd4: $basal\_mode \neq suspended$
Basal6.grd5: $t \in 0 \mathinner{..} c - 1$
Basal6.grd6: $rate\_setting2(t) = -1$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$

end

Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_return

when

Basal6.grd8: $prog\_basal = return\_get\_max$
Basal6.grd9: $add\_resume = 2$

then

Basal6.act1: $basal\_rate\_in := max\_value$
Basal6.act2: $basal\_mode := delivering$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $prog\_basal := null$
Basal6.act5: $add\_resume := 0$
act1: $prog := return\_basal\_resume$

end

Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call

when

grd1: $prog = call\_basal\_resume$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in = 0$
Basal6.grd3: ⟨theorem⟩ $basal\_mode = suspended$
Basal6.grd6: $add\_resume = 0$
Basal6.grd5: $prog\_basal = null$

then

Basal6.act1: $par\_get\_t := par\_basal\_resume\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_resume := 1$

end

Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_resume = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_resume := 2$

end

Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6·rate_update_return

when
Basal6.grd12: $add\_update = 1$
Basal6.grd4: $prog\_basal = return\_get\_min$
then
Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$
Basal6.act2: $btime := min\_value - par\_get\_t$
Basal6.act3: $add\_update := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_update$
end
Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.rate_update_call
when
grd1: $prog = call\_basal\_update$
Basal6.grd5: ⟨theorem⟩ $basal\_mode = delivering$
Basal6.grd7: ⟨theorem⟩ $rate\_setting2(par\_basal\_update\_rate\_t) \neq -1$
Basal6.grd3: $add\_update = 0$
Basal6.grd2: $prog\_basal = null$
then
Basal6.act1: $par\_get\_t := par\_basal\_update\_rate\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_update := 1$
end
Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_return
when
Basal6.grd8: $add\_start = 2$
Basal6.grd9: $prog\_basal = return\_get\_max$
then
Basal6.act1: $basal\_mode := delivering$
Basal6.act2: $basal\_rate\_in := max\_value$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $add\_start := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_start$
end
Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call
when
grd1: $prog = call\_basal\_start$
Basal6.grd3: $add\_start = 0$
Basal6.grd4: ⟨theorem⟩ $basal\_mode = stop$
Basal6.grd2: $prog\_basal = null$
then
Basal6.act1: $par\_get\_t := par\_basal\_start\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_start := 1$
end
Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call_2
when
Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_start = 1$
then
Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_start := 2$
end
Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$

extends Basal6.stop
when
Basal6.grd2: $prog\_basal = null$
Basal6.grd1: $basal\_mode = delivering$
grd1: $prog = call\_basal\_stop$
then
Basal6.act1: $basal\_mode := stop$
Basal6.act2: $basal\_rate\_in := 0$
act1: $prog := return\_basal\_stop$
end
Event Basal6.get_min_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_1
when
Basal6.grd4: $get\_min\_value\_add = 2$
Basal6.grd5: $par\_t = c$
then
Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_2
when
Basal6.grd5: $get\_min\_value\_add = 3$
then
Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_start
when
Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$
then
Basal6.act1: $par\_t := par\_get\_t + 1$
Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$
end
Event Basal6.find_min_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$
then
Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$
end
Event Basal6.find_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value_2
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$
then
Basal6.act1: $temp\_min := par\_t$

Basal6.act2: $get\_min\_value\_add := 3$

end

Event Basal6.get_max_value ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_max_value

when

Basal6.grd2: $get\_max\_value\_add = 2$

then

Basal6.act3: $prog\_basal := return\_get\_max$

Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$

Basal6.act2: $get\_max\_value\_add := 0$

end

Event Basal6.get_max_value_start ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_max_value_start

when

Basal6.grd2: $get\_max\_value\_add = 0$

Basal6.grd3: $prog\_basal = call\_get\_max$

then

Basal6.act1: $get\_max\_start\_t := par\_get\_t$

Basal6.act2: $get\_max\_value\_add := 1$

Basal6.act3: $par\_t\_max := par\_get\_t$

end

Event Basal6.get_max_value_1 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_max_value_1

when

Basal6.grd1: $get\_max\_value\_add = 1$

Basal6.grd3: $par\_t\_max \geq 0$

Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$

then

Basal6.act1: $par\_t\_max := par\_t\_max - 1$

end

Event Basal6.get_max_value_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_max_value_2

when

Basal6.grd1: $get\_max\_value\_add = 1$

Basal6.grd2: $par\_t\_max \geq 0$

Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$

then

Basal6.act1: $get\_max\_value\_add := 2$

end

END

MACHINE control_Basal6_NormalBolus_2
REFINES control_Basal6_NormalBolus
SEES c_normalbolus,c_prog
VARIABLES

rate_setting2
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
basal_rate_in
basal_mode
btime
par_basal_start_t
par_basal_resume_t
par_basal_update_rate_t
insulin_needed
normal_add
normal_delivering_time
normal_delivering_rate
normal_bolus_suspend
prog1
dmodule
d_update_time
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add
par_t_max
prog_basal
par_get_t
add_resume
add_update
add_start
prog

INVARIANTS

inv1: $prog1 \in PROG$
inv2: $prog1 = return\_normal\_finish \Rightarrow normal\_delivering\_rate = 0$

inv3: $prog1 = return\_normal\_suspend \Rightarrow normal\_delivering\_rate = 0$

inv4: $prog1 = return\_normal\_resume \Rightarrow normal\_delivering\_rate = 0$

inv5: $prog1 = return\_normal\_start \Rightarrow normal\_bolus\_work = FALSE \wedge normal\_suspend = FALSE \wedge normal\_add = 3$

inv6: $prog1 = return\_normal\_finish \Rightarrow normal\_suspend = FALSE \wedge time = t\_normal \wedge normal\_bolus\_work = TRUE$

inv7: $prog1 = return\_normal\_suspend \Rightarrow normal\_suspend = FALSE \wedge normal\_bolus\_work = TRUE$

inv8: $prog1 = return\_normal\_resume \Rightarrow normal\_suspend = TRUE$

inv9: $prog1 = call\_normal\_finish \Rightarrow normal\_suspend = FALSE \wedge time = t\_normal \wedge normal\_bolus\_work = TRUE$

inv10: $prog1 = call\_normal\_suspend \Rightarrow normal\_suspend = FALSE \wedge normal\_bolus\_work = TRUE$

inv11: $prog1 = call\_normal\_resume \Rightarrow normal\_suspend = TRUE$

inv12: $prog1 = call\_normal\_start \Rightarrow normal\_bolus\_work = FALSE \wedge normal\_bolus\_suspend = FALSE$

inv19: $normal\_add = 1 \vee normal\_add = 2 \Rightarrow prog1 = call\_normal\_start$

inv13: $prog1 = call\_normal\_suspend \Rightarrow normal\_add = 3 \wedge normal\_bolus\_suspend = FALSE$

inv14: $prog1 = call\_normal\_finish \Rightarrow normal\_bolus\_suspend = FALSE \wedge normal\_add = 3$

inv15: $prog1 = call\_normal\_resume \Rightarrow normal\_bolus\_suspend = TRUE \wedge normal\_add = 0$

inv20: $prog1 = call\_normal\_suspend \wedge normal\_suspend = FALSE \wedge normal\_bolus\_work = TRUE \Rightarrow normal\_add = 3$

inv21: $prog1 \in \{null, call\_basal\_start, return\_basal\_start, call\_basal\_stop, return\_basal\_stop, call\_basal\_suspend, return\_basal\_suspend, call\_basal\_resume, return\_basal\_resume, call\_basal\_update, return\_basal\_update\} \wedge normal\_suspend = FALSE \wedge normal\_bolus\_work = TRUE \Rightarrow normal\_add = 3$

inv22: $prog1 \in \{null, call\_basal\_start, return\_basal\_start, call\_basal\_stop, return\_basal\_stop, call\_basal\_suspend, return\_basal\_suspend, call\_basal\_resume, return\_basal\_resume, call\_basal\_update, return\_basal\_update\} \Rightarrow normal\_bolus\_suspend = normal\_suspend$

inv23: $prog1 = return\_normal\_suspend \Rightarrow normal\_bolus\_suspend = TRUE$

inv24: $prog1 = return\_normal\_resume \Rightarrow normal\_bolus\_suspend = FALSE$

inv25: $prog1 = return\_normal\_finish \Rightarrow normal\_bolus\_suspend = FALSE$

inv26: $prog1 \in \{call\_basal\_start, return\_basal\_start, call\_basal\_stop, return\_basal\_stop, call\_basal\_suspend, return\_basal\_suspend, call\_basal\_resume, return\_basal\_resume, call\_basal\_update, return\_basal\_update\} \Rightarrow prog = prog1$

inv27: $prog1 = null \Rightarrow prog = null$

inv28: $prog1 \in \{call\_normal\_start, return\_normal\_start, call\_normal\_suspend, return\_normal\_suspend, call\_normal\_finish, return\_normal\_finish, call\_normal\_resume, return\_normal\_resume\} \Rightarrow prog = null$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$

act2: $sd\_bolus\_work := FALSE$

act3: $sd\_preempted\_by\_normal := FALSE$

act7: $sd\_suspend := FALSE$

act8: $normal\_suspend := FALSE$

act9: $basal\_work := FALSE$

act10: $basal\_suspend := FALSE$

act11: $pump\_rate := 0$

act12: $basal\_rate := 0$

act13: $normal\_rate := 0$

act14: $sd\_rate := 0$

act15: $time := 0$

act16: $t\_basal := 0$

act17: $t\_normal := 0$

act18: $t\_sd := 0$

act19: $dmodule := FALSE$

act20: $d\_update\_time := 0$

Basal1.act4: $btime := c$
Basal1.act2: $basal\_rate\_in := 0$
Basal1.act3: $basal\_mode := stop$
Basal6.act15: $prog\_basal := null$
Basal6.act16: $par\_get\_t := 0$
Basal6.act17: $add\_resume := 0$
Basal6.act18: $add\_update := 0$
Basal6.act19: $add\_start := 0$
Basal6.act5: $rate\_setting2 := (1 \mathinner{..} c - 1 \times \{-1\}) \cup \{0 \mapsto 0\}$
Basal6.act6: $min\_value := 0$
Basal6.act7: $max\_value := 0$
Basal6.act11: $get\_min\_value\_add := 0$
Basal6.act8: $par\_t := 0$
Basal6.act9: $temp\_min := 0$
Basal6.act10: $get\_min\_start\_t := 0$
Basal6.act12: $get\_max\_start\_t := 0$
Basal6.act13: $get\_max\_value\_add := 0$
Basal6.act14: $par\_t\_max := 0$
act21: $prog := null$
act22: $par\_basal\_start\_t := 0$
act23: $par\_basal\_resume\_t := 0$
act24: $par\_basal\_update\_rate\_t := 0$
NormalBolus.act1: $insulin\_needed := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_delivering\_rate := 0$
NormalBolus.act4: $normal\_add := 0$
NormalBolus.act5: $normal\_bolus\_suspend := FALSE$
act31: $prog1 := null$

end

Event control5·normal_bolus_start_1_return ⟨ordinary⟩ $\widehat{=}$

refines control5.normal_bolus_start_1

any

t2

where

grd6: $normal\_bolus\_work = FALSE$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \land time \neq d\_update\_time$
control5.grd2: $sd\_bolus\_work = TRUE$
control5.grd3: $sd\_preempted\_by\_normal = FALSE$
control5.grd4: $sd\_suspend = FALSE$
grd1: $prog1 = return\_normal\_start$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

with

t: $t = normal\_delivering\_time$
r: $r = normal\_delivering\_rate$

then

control5.act6: $t\_normal := time + normal\_delivering\_time$
control5.act7: $t\_sd := t\_sd - time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $sd\_preempted\_by\_normal := TRUE$
control5.act3: $normal\_rate := normal\_delivering\_rate$
control5.act4: $sd\_rate := 0$
control5.act5: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act8: $d\_update\_time := t2$
act1: $prog1 := null$

end

Event control5·normal_bolus_start_1_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog1 = null$
grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$

then

act1: $prog1 := call\_normal\_start$

end

Event control5.normal_bolus_start_2_return ⟨ordinary⟩ $\widehat{=}$

refines control5.normal_bolus_start_2

when

control5.grd2: $sd\_bolus\_work = FALSE$
grd1: $prog1 = return\_normal\_start$

with

t: $t = normal\_delivering\_time$
r: $r = normal\_delivering\_rate$

then

control5·act4: $t\_normal := time + normal\_delivering\_time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act1: $prog1 := null$

end

Event control5·normal_bolus_start_2_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog1 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = FALSE$
grd4: $normal\_suspend = FALSE$

then

act1: $prog1 := call\_normal\_start$

end

Event control5·normal_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$

refines control5.normal_bolus_finish

when

grd1: $prog1 = return\_normal\_finish$

then

control5.act4: $t\_normal := 0$
control5.act1: $normal\_bolus\_work := FALSE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$

end

Event control5·normal_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog1 = null$
grd2: $time = t\_normal$
grd3: $normal\_bolus\_work = TRUE$
grd4: $normal\_suspend = FALSE$

then

act1: $prog1 := call\_normal\_finish$

end

Event control5·normal_suspend_return ⟨ordinary⟩ $\widehat{=}$

refines control5.normal_suspend

when

grd1: $prog1 = return\_normal\_suspend$

then

control5.act4: $t\_normal := t\_normal - time$
control5.act1: $normal\_suspend := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$

control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$

end

Event control5·normal_bolus_suspend_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog1 = null$
grd2: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$

then

act1: $prog1 := call\_normal\_suspend$

end

Event control5·normal_resume_return ⟨ordinary⟩ $\widehat{=}$

refines control5.normal_resume

when

grd1: $prog1 = return\_normal\_resume$

then

control5.act4: $t\_normal := 0$
control5.act1: $normal\_suspend := FALSE$
control5.act2: $normal\_rate := 0$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
act2: $normal\_bolus\_work := FALSE$

end

Event control5·normal_bolus_resume_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog1 = null$
grd2: $normal\_suspend = TRUE$

then

act1: $prog1 := call\_normal\_resume$

end

Event control5·square_or_dual_bolus_start_s ⟨ordinary⟩ $\widehat{=}$

extends control5·square_or_dual_bolus_start_s

any

r
t

where

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd4: $r \in \mathbb{N}_1$
grd5: $t \in \mathbb{N}_1$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$
act5: $dmodule := FALSE$

end

Event control5·square_or_dual_bolus_start_d ⟨ordinary⟩ $\widehat{=}$

extends control5·square_or_dual_bolus_start_d

any

r
t
$t0$ t: both bolus, t0:for normal bolus

where

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$

grd4: $r \in \mathbb{N}_1$
grd5: $t \in \mathbb{N}_1$
grd6: $t0 \in \mathbb{N}_1$
grd7: $t > t0$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$
act5: $dmodule := TRUE$
act6: $d\_update\_time := time + t0$

end

Event control5.square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_bolus_finish

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $dmodule := FALSE$

end

Event control5.square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_bolus_resume_from_normal

any

r
$t2$

where

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd7: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $d\_update\_time := t2$

end

Event control5.sd_suspend ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_suspend

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $sd\_suspend := TRUE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$

act6: $d\_update\_time := 0$

end

Event control5.square_or_dual_update_rate ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_update_rate

any

r

where

grd2: $sd\_suspend = FALSE$

grd3: $sd\_bolus\_work = TRUE$

grd4: $sd\_preempted\_by\_normal = FALSE$

grd5: $r \in \mathbb{N}_1$

grd6: $dmodule = TRUE$

grd7: $time = d\_update\_time$

then

act1: $sd\_rate := r$

act2: $pump\_rate := r + basal\_rate$

act3: $dmodule := FALSE$

act4: $d\_update\_time := 0$

end

Event control5.sd_resume ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_resume

when

grd1: $sd\_suspend = TRUE$

then

act1: $sd\_bolus\_work := FALSE$

act2: $sd\_suspend := FALSE$

act3: $sd\_rate := 0$

act4: $pump\_rate := basal\_rate$

act5: $t\_sd := 0$

end

Event control5·basal_start_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_return

when

grd1: $prog = return\_basal\_start$

grd2: $prog1 = return\_basal\_start$

then

control5.act4: $t\_basal := time + btime$

control5.act1: $basal\_work := TRUE$

control5.act2: $basal\_rate := basal\_rate\_in$

control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$

act1: $prog := null$

act2: $prog1 := null$

end

Event control5·basal_start_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_call

when

grd1: $basal\_work = FALSE$

grd2: $basal\_suspend = FALSE$

grd3: $prog = null$

grd4: $prog1 = null$

then

act1: $prog := call\_basal\_start$

act2: $par\_basal\_start\_t := timemodc$

act3: $prog1 := call\_basal\_start$

end

Event control5·basal_stop_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_stop_return

when
grd1: $prog = return\_basal\_stop$
grd2: $prog1 = return\_basal\_stop$
then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_work := FALSE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
end

Event control5·basal_stop_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_stop_call
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd3: $prog = null$
grd4: $prog1 = null$
then
act1: $prog := call\_basal\_stop$
act2: $prog1 := call\_basal\_stop$
end

Event control5·basal_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_suspend_return
when
grd1: $prog = return\_basal\_suspend$
grd2: $prog1 = return\_basal\_suspend$
then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_suspend := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
end

Event control5·basal_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_suspend_call
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd4: $basal\_rate \neq 0$
grd3: $prog = null$
grd5: $prog1 = null$
then
act1: $prog := call\_basal\_suspend$
act2: $prog1 := call\_basal\_suspend$
end

Event control5·basal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_resume_return
when
grd1: $prog = return\_basal\_resume$
grd2: $prog1 = return\_basal\_resume$
then
control5.act1: $basal\_suspend := FALSE$
control5.act4: $t\_basal := btime + time$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$

act1: $prog := null$
act2: $prog1 := null$

end

Event control5·basal_resume_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_resume_call

when

grd1: $basal\_suspend = TRUE$
grd2: $prog = null$
grd3: $prog1 = null$

then

act1: $prog := call\_basal\_resume$
act2: $par\_basal\_resume\_t := timemodc$
act3: $prog1 := call\_basal\_resume$

end

Event control5·basal_update_rate_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·basal_update_rate_return

when

grd1: $prog = return\_basal\_update$
grd2: $prog1 = return\_basal\_update$

then

control5.act3: $t\_basal := time + btime$
control5.act1: $basal\_rate := basal\_rate\_in$
control5.act2: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$

end

Event control5·basal_update_rate_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_update_rate_call

when

grd1: $t\_basal = time$
grd2: $basal\_suspend = FALSE$
grd3: $basal\_work = TRUE$
grd4: $prog = null$
grd5: $t\_basal \in dom(rate\_setting2 \rhd \{-1\})$
grd6: $prog1 = null$

then

act1: $prog := call\_basal\_update$
act2: $par\_basal\_update\_rate\_t := t\_basal$
act3: $prog1 := call\_basal\_update$

end

Event control5.timer ⟨ordinary⟩ $\widehat{=}$
extends control5.timer

when

grd1:
$\neg($
$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$
$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$
$(time = t\_sd)) \lor$
$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$
$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$
$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$
$)$

then

act1: $time := time + 1$

end

Event NormalBolus.normal_bolus_start_calculate_insulin_needed ⟨ordinary⟩ $\widehat{=}$
refines NormalBolus.normal_bolus_start_calculate_insulin_needed

any
insulin
where
NormalBolus.grd1: $insulin > 0$
NormalBolus.grd3: $normal\_add = 0$
grd1: $prog1 = call\_normal\_start$
then
NormalBolus.act1: $insulin\_needed := insulin$
NormalBolus.act2: $normal\_add := 1$
end

Event NormalBolus.normal_bolus_start_calculate_lasting_time ⟨ordinary⟩ $\widehat{=}$
extends NormalBolus.normal_bolus_start_calculate_lasting_time
when
NormalBolus.grd1: $normal\_add = 1$
then
NormalBolus.act1: $normal\_delivering\_time := insulin\_needed/normal\_bolus\_rate$
NormalBolus.act2: $insulin\_needed := 0$
NormalBolus.act3: $normal\_add := 2$
end

Event NormalBolus.normal_bolus_delivery ⟨ordinary⟩ $\widehat{=}$
extends NormalBolus.normal_bolus_delivery
when
NormalBolus.grd2: $normal\_add = 2$
then
NormalBolus.act1: $normal\_delivering\_rate := normal\_bolus\_rate$
NormalBolus.act2: $normal\_add := 3$
act1: $prog1 := return\_normal\_start$
end

Event NormalBolus.normal_bolus_suspend ⟨ordinary⟩ $\widehat{=}$
refines NormalBolus.normal_bolus_suspend
when
grd1: $prog1 = call\_normal\_suspend$
then
NormalBolus.act1: $normal\_delivering\_rate := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_add := 0$
NormalBolus.act4: $normal\_bolus\_suspend := TRUE$
act1: $prog1 := return\_normal\_suspend$
end

Event NormalBolus.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$
refines NormalBolus.normal_bolus_finish
when
grd1: $prog1 = call\_normal\_finish$
then
NormalBolus.act1: $normal\_delivering\_rate := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_add := 0$
act1: $prog1 := return\_normal\_finish$
end

Event NormalBolus.normal_bolus_resume ⟨ordinary⟩ $\widehat{=}$
refines NormalBolus ·normal_bolus_resume
when
grd1: $prog1 = call\_normal\_resume$
then
NormalBolus.act1: $normal\_bolus\_suspend := FALSE$
act1: $normal\_delivering\_rate := 0$
act2: $prog1 := return\_normal\_resume$
end

Event Basal6·basal_suspend ⟨ordinary⟩ $\widehat{=}$
extends Basal6.basal_suspend
when
grd1: $prog = call\_basal\_suspend$
Basal6.grd3: $prog\_basal = null$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in \neq 0$
Basal6.grd2: ⟨theorem⟩ $basal\_mode = delivering$
grd2: $prog1 = call\_basal\_suspend$
then
Basal6.act1: $basal\_rate\_in := 0$
Basal6.act2: $basal\_mode := suspended$
act1: $prog := return\_basal\_suspend$
act2: $prog1 := return\_basal\_suspend$
end
Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6·change_setting
any
t
r
where
Basal6.grd5: $prog\_basal = null$
Basal6.grd6: $t \in 0 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$
then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$
end
Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6.delete_setting
any
t
where
Basal6.grd5: $prog\_basal = null$
Basal6.grd2: $basal\_mode \neq suspended$
Basal6.grd6: $t \in 1 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
grd1: $t \neq par\_basal\_update\_rate\_t$
then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto -1\}$
end
Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6.add_setting
any
t
r
where
Basal6.grd9: $prog\_basal = null$
Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$
Basal6.grd4: $basal\_mode \neq suspended$
Basal6.grd5: $t \in 0 \mathinner{..} c - 1$
Basal6.grd6: $rate\_setting2(t) = -1$
then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$
end
Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends Basal6.basal_resume_return
when
Basal6.grd8: $prog\_basal = return\_get\_max$

Basal6.grd9: $add\_resume = 2$

then

Basal6.act1: $basal\_rate\_in := max\_value$
Basal6.act2: $basal\_mode := delivering$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $prog\_basal := null$
Basal6.act5: $add\_resume := 0$
act1: $prog := return\_basal\_resume$
act2: $prog1 := return\_basal\_resume$

end

Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call

when

grd1: $prog = call\_basal\_resume$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in = 0$
Basal6.grd3: ⟨theorem⟩ $basal\_mode = suspended$
Basal6.grd6: $add\_resume = 0$
Basal6.grd5: $prog\_basal = null$
grd2: $prog1 = call\_basal\_resume$

then

Basal6.act1: $par\_get\_t := par\_basal\_resume\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_resume := 1$

end

Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_resume = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_resume := 2$

end

Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6·rate_update_return

when

Basal6.grd12: $add\_update = 1$
Basal6.grd4: $prog\_basal = return\_get\_min$

then

Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$
Basal6.act2: $btime := min\_value - par\_get\_t$
Basal6.act3: $add\_update := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_update$
act6: $prog1 := return\_basal\_update$

end

Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.rate_update_call

when

grd1: $prog = call\_basal\_update$
Basal6.grd5: ⟨theorem⟩ $basal\_mode = delivering$
Basal6.grd7: ⟨theorem⟩ $rate\_setting2(par\_basal\_update\_rate\_t) \neq -1$
Basal6.grd3: $add\_update = 0$
Basal6.grd2: $prog\_basal = null$
grd2: $prog1 = call\_basal\_update$

then

Basal6.act1: $par\_get\_t := par\_basal\_update\_rate\_t$
Basal6.act2: $prog\_basal := call\_get\_min$

Basal6.act3: $add\_update := 1$

end

Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_return

when

Basal6.grd8: $add\_start = 2$

Basal6.grd9: $prog\_basal = return\_get\_max$

then

Basal6.act1: $basal\_mode := delivering$

Basal6.act2: $basal\_rate\_in := max\_value$

Basal6.act3: $btime := min\_value - par\_get\_t$

Basal6.act4: $add\_start := 0$

act4: $prog\_basal := null$

act5: $prog := return\_basal\_start$

act6: $prog1 := return\_basal\_start$

end

Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_call

when

grd1: $prog = call\_basal\_start$

Basal6.grd3: $add\_start = 0$

Basal6.grd4: ⟨theorem⟩ $basal\_mode = stop$

Basal6.grd2: $prog\_basal = null$

grd2: $prog1 = call\_basal\_start$

then

Basal6.act1: $par\_get\_t := par\_basal\_start\_t$

Basal6.act2: $prog\_basal := call\_get\_min$

Basal6.act3: $add\_start := 1$

end

Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$

Basal6.grd2: $add\_start = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$

Basal6.act2: $add\_start := 2$

end

Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$

extends Basal6.stop

when

Basal6.grd2: $prog\_basal = null$

Basal6.grd1: $basal\_mode = delivering$

grd1: $prog = call\_basal\_stop$

grd2: $prog1 = call\_basal\_stop$

then

Basal6.act1: $basal\_mode := stop$

Basal6.act2: $basal\_rate\_in := 0$

act1: $prog := return\_basal\_stop$

act2: $prog1 := return\_basal\_stop$

end

Event Basal6.get_min_value_1 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_min_value_1

when

Basal6.grd4: $get\_min\_value\_add = 2$

Basal6.grd5: $par\_t = c$

then

Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$

end

Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_2

when
Basal6.grd5: $get\_min\_value\_add = 3$

then
Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$

end

Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_start

when
Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$

then
Basal6.act1: $par\_t := par\_get\_t + 1$
Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$

end

Event Basal6.find_min_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value

when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$

then
Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$

end

Event Basal6.find_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value_2

when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$

then
Basal6.act1: $temp\_min := par\_t$
Basal6.act2: $get\_min\_value\_add := 3$

end

Event Basal6.get_max_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value

when
Basal6.grd2: $get\_max\_value\_add = 2$

then
Basal6.act3: $prog\_basal := return\_get\_max$
Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$
Basal6.act2: $get\_max\_value\_add := 0$

end

Event Basal6.get_max_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_start

when
Basal6.grd2: $get\_max\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_max$

then

Basal6.act1: $get\_max\_start\_t := par\_get\_t$

Basal6.act2: $get\_max\_value\_add := 1$

Basal6.act3: $par\_t\_max := par\_get\_t$

end

Event Basal6·get_max_value_1 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_max_value_1

when

Basal6.grd1: $get\_max\_value\_add = 1$

Basal6.grd3: $par\_t\_max \geq 0$

Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$

then

Basal6.act1: $par\_t\_max := par\_t\_max - 1$

end

Event Basal6.get_max_value_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_max_value_2

when

Basal6.grd1: $get\_max\_value\_add = 1$

Basal6.grd2: $par\_t\_max \geq 0$

Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$

then

Basal6.act1: $get\_max\_value\_add := 2$

end

END

MACHINE control_Basal6_NormalBolus_2_Square_Dual_bolus2
REFINES control_Basal6_NormalBolus_2
SEES c_normalbolus,c_prog,c_sd_bolus
VARIABLES

rate_setting2
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
basal_rate_in
basal_mode
btime
par_basal_start_t
par_basal_resume_t
par_basal_update_rate_t
insulin_needed
normal_add
normal_delivering_time
normal_delivering_rate
normal_bolus_suspend
dmodule
d_update_time
prog1
state
s_r
s_t
d_deliver_time
d_deliver_rate
d_t
sd_module
sd_flag
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add
par_t_max
prog_basal

par_get_t

add_resume

add_update

add_start

prog

INVARIANTS

Square_Dual_bolus2.inv1: $d\_deliver\_time \in \mathbb{N}$

Square_Dual_bolus2.inv2: $d\_deliver\_rate \in \mathbb{N}$

Square_Dual_bolus2.inv3: $d\_t \in \mathbb{N}$

inv1: $s\_r \in \mathbb{N}$

inv2: $s\_t \in \mathbb{N}$

Square_Dual_bolus2.inv4: $sd\_module \in SDF$

Square_Dual_bolus2.inv5: $sd\_flag \in SDF$

Square_Dual_bolus2.inv6: $state = off \lor state = suspend \Rightarrow d\_deliver\_time = 0 \land d\_deliver\_rate = 0$

Square_Dual_bolus2.inv7: $state = deliver \Rightarrow d\_deliver\_time \geq 0 \land d\_deliver\_rate > 0$

Square_Dual_bolus2.inv8: $state = off \lor state = suspend \Rightarrow d\_t = 0$

Square_Dual_bolus2.inv9: $state = deliver \lor state = preempt \Rightarrow s\_r > 0$

Square_Dual_bolus2.inv10: $state = preempt \Rightarrow d\_deliver\_time \geq 0 \land d\_deliver\_rate = 0$

Square_Dual_bolus2.inv11: $sd\_module = d \land sd\_flag = s \land state = deliver \Rightarrow d\_deliver\_rate = s\_r$

Square_Dual_bolus2.inv12: $sd\_module = d \land sd\_flag = d \land state = deliver \Rightarrow d\_deliver\_rate = normal\_bolus\_rate$

Square_Dual_bolus2.inv13: $sd\_module = s \land state = deliver \Rightarrow d\_deliver\_rate = s\_r$

Square_Dual_bolus2.inv14: $state = off \lor state = suspend \Rightarrow sd\_flag = d$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$

act2: $sd\_bolus\_work := FALSE$

act3: $sd\_preempted\_by\_normal := FALSE$

act7: $sd\_suspend := FALSE$

act8: $normal\_suspend := FALSE$

act9: $basal\_work := FALSE$

act10: $basal\_suspend := FALSE$

act11: $pump\_rate := 0$

act12: $basal\_rate := 0$

act13: $normal\_rate := 0$

act14: $sd\_rate := 0$

act15: $time := 0$

act16: $t\_basal := 0$

act17: $t\_normal := 0$

act18: $t\_sd := 0$

act19: $dmodule := FALSE$

act20: $d\_update\_time := 0$

Basal1.act4: $btime := c$

Basal1.act2: $basal\_rate\_in := 0$

Basal1.act3: $basal\_mode := stop$

Basal6.act15: $prog\_basal := null$

Basal6.act16: $par\_get\_t := 0$

Basal6.act17: $add\_resume := 0$

Basal6.act18: $add\_update := 0$

Basal6.act19: $add\_start := 0$

Basal6.act5: $rate\_setting2 := (1 \mathinner{..} c - 1 \times \{-1\}) \cup \{0 \mapsto 0\}$

Basal6.act6: $min\_value := 0$

Basal6.act7: $max\_value := 0$

Basal6.act11: $get\_min\_value\_add := 0$

Basal6.act8: $par\_t := 0$

Basal6.act9: $temp\_min := 0$
Basal6.act10: $get\_min\_start\_t := 0$
Basal6.act12: $get\_max\_start\_t := 0$
Basal6.act13: $get\_max\_value\_add := 0$
Basal6.act14: $par\_t\_max := 0$
act21: $prog := null$
act22: $par\_basal\_start\_t := 0$
act23: $par\_basal\_resume\_t := 0$
act24: $par\_basal\_update\_rate\_t := 0$
NormalBolus.act1: $insulin\_needed := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_delivering\_rate := 0$
NormalBolus.act4: $normal\_add := 0$
NormalBolus.act5: $normal\_bolus\_suspend := FALSE$
act31: $prog1 := null$
Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act2: $s\_r := 0$
Square_Dual_bolus2.act3: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_module := s$
Square_Dual_bolus2.act10: $sd\_flag := d$

end

Event control5·normal_bolus_start_1_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_1_return

any
$t2$
where
grd6: $normal\_bolus\_work = FALSE$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \land time \neq d\_update\_time$
control5.grd2: $sd\_bolus\_work = TRUE$
control5.grd3: $sd\_preempted\_by\_normal = FALSE$
control5.grd4: $sd\_suspend = FALSE$
grd1: $prog1 = return\_normal\_start$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$
then
control5.act6: $t\_normal := time + normal\_delivering\_time$
control5.act7: $t\_sd := t\_sd - time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $sd\_preempted\_by\_normal := TRUE$
control5.act3: $normal\_rate := normal\_delivering\_rate$
control5.act4: $sd\_rate := 0$
control5.act5: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act8: $d\_update\_time := t2$
act1: $prog1 := null$

end

Event control5·normal_bolus_start_1_call ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_1_call

when
grd1: $prog1 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
then
act1: $prog1 := call\_normal\_start$

end

Event control5.normal_bolus_start_2_return ⟨ordinary⟩ $\widehat{=}$

extends control5.normal_bolus_start_2_return
when
control5.grd2: $sd\_bolus\_work = FALSE$
grd1: $prog1 = return\_normal\_start$
then
control5·act4: $t\_normal := time + normal\_delivering\_time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act1: $prog1 := null$
end
Event control5·normal_bolus_start_2_call ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_2_call
when
grd1: $prog1 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = FALSE$
grd4: $normal\_suspend = FALSE$
then
act1: $prog1 := call\_normal\_start$
end
Event control5·normal_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_finish_return
when
grd1: $prog1 = return\_normal\_finish$
then
control5.act4: $t\_normal := 0$
control5.act1: $normal\_bolus\_work := FALSE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
end
Event control5·normal_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$
extends control5 ·normal_bolus_finish_call
when
grd1: $prog1 = null$
grd2: $time = t\_normal$
grd3: $normal\_bolus\_work = TRUE$
grd4: $normal\_suspend = FALSE$
then
act1: $prog1 := call\_normal\_finish$
end
Event control5·normal_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_suspend_return
when
grd1: $prog1 = return\_normal\_suspend$
then
control5.act4: $t\_normal := t\_normal - time$
control5.act1: $normal\_suspend := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
end
Event control5·normal_bolus_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_suspend_call
when
grd1: $prog1 = null$

grd2: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$

then

act1: $prog1 := call\_normal\_suspend$

end

Event control5·normal_resume_return ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_resume_return

when

grd1: $prog1 = return\_normal\_resume$

then

control5.act4: $t\_normal := 0$
control5.act1: $normal\_suspend := FALSE$
control5.act2: $normal\_rate := 0$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
act2: $normal\_bolus\_work := FALSE$

end

Event control5·normal_bolus_resume_call ⟨ordinary⟩ $\widehat{=}$

extends control5 ·normal_bolus_resume_call

when

grd1: $prog1 = null$
grd2: $normal\_suspend = TRUE$

then

act1: $prog1 := call\_normal\_resume$

end

Event control5·square_or_dual_bolus_start_s ⟨ordinary⟩ $\widehat{=}$

extends control5·square_or_dual_bolus_start_s

any

r
t

where

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd4: $r \in \mathbb{N}_1$
grd5: $t \in \mathbb{N}_1$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$
act5: $dmodule := FALSE$

end

Event control5·square_or_dual_bolus_start_d ⟨ordinary⟩ $\widehat{=}$

extends control5·square_or_dual_bolus_start_d

any

r
t
$t0$ t: both bolus, t0:for normal bolus

where

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd4: $r \in \mathbb{N}_1$
grd5: $t \in \mathbb{N}_1$
grd6: $t0 \in \mathbb{N}_1$
grd7: $t > t0$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t$
act5: $dmodule := TRUE$
act6: $d\_update\_time := time + t0$

end

Event control5.square_or_dual_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_finish

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $dmodule := FALSE$

end

Event control5.square_or_dual_bolus_resume_from_normal ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_resume_from_normal

any

r
$t2$

where

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $r \in \mathbb{N}_1$
grd6: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd7: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := r$
act3: $pump\_rate := r + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $d\_update\_time := t2$

end

Event control5.sd_suspend ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_suspend

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $sd\_suspend := TRUE$
act2: $sd\_rate := 0$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$
act6: $d\_update\_time := 0$

end

Event control5.square_or_dual_update_rate ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_update_rate

any

r

where

grd2: $sd\_suspend = FALSE$

grd3: $sd\_bolus\_work = TRUE$

grd4: $sd\_preempted\_by\_normal = FALSE$

grd5: $r \in \mathbb{N}_1$

grd6: $dmodule = TRUE$

grd7: $time = d\_update\_time$

then

act1: $sd\_rate := r$

act2: $pump\_rate := r + basal\_rate$

act3: $dmodule := FALSE$

act4: $d\_update\_time := 0$

end

Event control5.sd_resume ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_resume

when

grd1: $sd\_suspend = TRUE$

then

act1: $sd\_bolus\_work := FALSE$

act2: $sd\_suspend := FALSE$

act3: $sd\_rate := 0$

act4: $pump\_rate := basal\_rate$

act5: $t\_sd := 0$

end

Event control5·basal_start_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_return

when

grd1: $prog = return\_basal\_start$

grd2: $prog1 = return\_basal\_start$

then

control5.act4: $t\_basal := time + btime$

control5.act1: $basal\_work := TRUE$

control5.act2: $basal\_rate := basal\_rate\_in$

control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$

act1: $prog := null$

act2: $prog1 := null$

end

Event control5·basal_start_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_call

when

grd1: $basal\_work = FALSE$

grd2: $basal\_suspend = FALSE$

grd3: $prog = null$

grd4: $prog1 = null$

then

act1: $prog := call\_basal\_start$

act2: $par\_basal\_start\_t := timemodc$

act3: $prog1 := call\_basal\_start$

end

Event control5·basal_stop_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_stop_return

when

grd1: $prog = return\_basal\_stop$

grd2: $prog1 = return\_basal\_stop$

then

control5.act4: $t\_basal := 0$

control5.act1: $basal\_work := FALSE$

control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$

end

Event control5·basal_stop_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_stop_call

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd3: $prog = null$
grd4: $prog1 = null$

then

act1: $prog := call\_basal\_stop$
act2: $prog1 := call\_basal\_stop$

end

Event control5·basal_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_suspend_return

when

grd1: $prog = return\_basal\_suspend$
grd2: $prog1 = return\_basal\_suspend$

then

control5.act4: $t\_basal := 0$
control5.act1: $basal\_suspend := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$

end

Event control5·basal_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_suspend_call

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

grd4: $basal\_rate \neq 0$
grd3: $prog = null$
grd5: $prog1 = null$

then

act1: $prog := call\_basal\_suspend$
act2: $prog1 := call\_basal\_suspend$

end

Event control5·basal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_resume_return

when

grd1: $prog = return\_basal\_resume$
grd2: $prog1 = return\_basal\_resume$

then

control5.act1: $basal\_suspend := FALSE$
control5.act4: $t\_basal := btime + time$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$

end

Event control5·basal_resume_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_resume_call

when

grd1: $basal\_suspend = TRUE$

grd2: $prog = null$

grd3: $prog1 = null$

then

act1: $prog := call\_basal\_resume$

act2: $par\_basal\_resume\_t := timemodc$

act3: $prog1 := call\_basal\_resume$

end

Event control5·basal_update_rate_return ⟨ordinary⟩ $\widehat{=}$

extends control5 ·basal_update_rate_return

when

grd1: $prog = return\_basal\_update$

grd2: $prog1 = return\_basal\_update$

then

control5.act3: $t\_basal := time + btime$

control5.act1: $basal\_rate := basal\_rate\_in$

control5.act2: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$

act1: $prog := null$

act2: $prog1 := null$

end

Event control5·basal_update_rate_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_update_rate_call

when

grd1: $t\_basal = time$

grd2: $basal\_suspend = FALSE$

grd3: $basal\_work = TRUE$

grd4: $prog = null$

grd5: $t\_basal \in dom(rate\_setting2 \rhd \{-1\})$

grd6: $prog1 = null$

then

act1: $prog := call\_basal\_update$

act2: $par\_basal\_update\_rate\_t := t\_basal$

act3: $prog1 := call\_basal\_update$

end

Event control5.timer ⟨ordinary⟩ $\widehat{=}$

extends control5.timer

when

grd1:

$\neg($

$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$

$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$

$(time = t\_sd)) \lor$

$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$

$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$

$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$

$)$

then

act1: $time := time + 1$

end

Event NormalBolus.normal_bolus_start_calculate_insulin_needed ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_start_calculate_insulin_needed

any

$insulin$

where

NormalBolus.grd1: $insulin > 0$

NormalBolus.grd3: $normal\_add = 0$

grd1: $prog1 = call\_normal\_start$

then
NormalBolus.act1: $insulin\_needed := insulin$
NormalBolus.act2: $normal\_add := 1$
end
Event NormalBolus.normal_bolus_start_calculate_lasting_time ⟨ordinary⟩ $\widehat{=}$
extends NormalBolus.normal_bolus_start_calculate_lasting_time
when
NormalBolus.grd1: $normal\_add = 1$
then
NormalBolus.act1: $normal\_delivering\_time := insulin\_needed / normal\_bolus\_rate$
NormalBolus.act2: $insulin\_needed := 0$
NormalBolus.act3: $normal\_add := 2$
end
Event NormalBolus.normal_bolus_delivery ⟨ordinary⟩ $\widehat{=}$
extends NormalBolus.normal_bolus_delivery
when
NormalBolus.grd2: $normal\_add = 2$
then
NormalBolus.act1: $normal\_delivering\_rate := normal\_bolus\_rate$
NormalBolus.act2: $normal\_add := 3$
act1: $prog1 := return\_normal\_start$
end
Event NormalBolus.normal_bolus_suspend ⟨ordinary⟩ $\widehat{=}$
extends NormalBolus.normal_bolus_suspend
when
grd1: $prog1 = call\_normal\_suspend$
then
NormalBolus.act1: $normal\_delivering\_rate := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_add := 0$
NormalBolus.act4: $normal\_bolus\_suspend := TRUE$
act1: $prog1 := return\_normal\_suspend$
end
Event NormalBolus.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$
extends NormalBolus.normal_bolus_finish
when
grd1: $prog1 = call\_normal\_finish$
then
NormalBolus.act1: $normal\_delivering\_rate := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_add := 0$
act1: $prog1 := return\_normal\_finish$
end
Event NormalBolus.normal_bolus_resume ⟨ordinary⟩ $\widehat{=}$
extends NormalBolus.normal_bolus_resume
when
grd1: $prog1 = call\_normal\_resume$
then
NormalBolus.act1: $normal\_bolus\_suspend := FALSE$
act1: $normal\_delivering\_rate := 0$
act2: $prog1 := return\_normal\_resume$
end
Event Square_Dual_bolus2.start ⟨ordinary⟩ $\widehat{=}$
any
t
r
where

Square_Dual_bolus2.grd1: $state = off$
Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$
Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$

then

Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act2: $s\_r := r$
Square_Dual_bolus2.act3: $s\_t := t$
Square_Dual_bolus2.act6: $sd\_module := s$
Square_Dual_bolus2.act7: $d\_deliver\_time := t$
Square_Dual_bolus2.act8: $d\_deliver\_rate := r$

end

Event Square_Dual_bolus2.start_dual ⟨ordinary⟩ $\widehat{=}$

any

t
r
td

where

Square_Dual_bolus2.grd1: $state = off$
Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$
Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$
Square_Dual_bolus2.grd4: $td \in \mathbb{N}_1$

then

Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act2: $s\_r := r$
Square_Dual_bolus2.act3: $s\_t := t$
Square_Dual_bolus2.act6: $d\_deliver\_time := t + td$
Square_Dual_bolus2.act7: $d\_deliver\_rate := normal\_bolus\_rate$
Square_Dual_bolus2.act8: $d\_t := td$
Square_Dual_bolus2.act9: $sd\_module := d$

end

Event Square_Dual_bolus2.update_to_dual ⟨ordinary⟩ $\widehat{=}$

when

Square_Dual_bolus2.grd2: $state = deliver$
Square_Dual_bolus2.grd3: $sd\_module = d$
Square_Dual_bolus2.grd4: $sd\_flag = d$

then

Square_Dual_bolus2.act2: $d\_deliver\_rate := s\_r$
Square_Dual_bolus2.act3: $sd\_flag := s$

end

Event Square_Dual_bolus2.finish ⟨ordinary⟩ $\widehat{=}$

when

Square_Dual_bolus2.grd1: $state = deliver$
Square_Dual_bolus2.grd2: $sd\_module = d \Rightarrow sd\_flag = s$

then

Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act4: $s\_r := 0$
Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$

end

Event Square_Dual_bolus2.suspend ⟨ordinary⟩ $\widehat{=}$

when

Square_Dual_bolus2.grd1: $state = deliver$

then

Square_Dual_bolus2.act1: $state := suspend$
Square_Dual_bolus2.act4: $s\_r := 0$
Square_Dual_bolus2.act5: $s\_t := 0$

Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$

end

Event Square_Dual_bolus2.resume ⟨ordinary⟩ $\widehat{=}$

when

Square_Dual_bolus2.grd1: $state = suspend$

then

Square_Dual_bolus2.act1: $state := off$

end

Event Square_Dual_bolus2.preempted ⟨ordinary⟩ $\widehat{=}$

any

t time left for square bolus

where

Square_Dual_bolus2.grd1: $state = deliver$
Square_Dual_bolus2.grd2: $t \in 0 \mathinner{..} d\_deliver\_time$

then

Square_Dual_bolus2.act1: $state := preempt$
Square_Dual_bolus2.act4: $d\_deliver\_time := t$
Square_Dual_bolus2.act5: $d\_deliver\_rate := 0$

end

Event Square_Dual_bolus2.resume_from_preempt ⟨ordinary⟩ $\widehat{=}$

any

r

where

Square_Dual_bolus2.grd1: $state = preempt$
Square_Dual_bolus2.grd2: $sd\_module = s \Rightarrow r = s\_r$
Square_Dual_bolus2.grd3: $sd\_module = d \wedge sd\_flag = d \Rightarrow r = normal\_bolus\_rate$
Square_Dual_bolus2.grd4: $sd\_module = d \wedge sd\_flag = s \Rightarrow r = s\_r$

then

Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act4: $d\_deliver\_rate := r$

end

Event Basal6·basal_suspend ⟨ordinary⟩ $\widehat{=}$

extends Basal6·basal_suspend

when

grd1: $prog = call\_basal\_suspend$
Basal6.grd3: $prog\_basal = null$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in \neq 0$
Basal6.grd2: ⟨theorem⟩ $basal\_mode = delivering$
grd2: $prog1 = call\_basal\_suspend$

then

Basal6.act1: $basal\_rate\_in := 0$
Basal6.act2: $basal\_mode := suspended$
act1: $prog := return\_basal\_suspend$
act2: $prog1 := return\_basal\_suspend$

end

Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6·change_setting

any

t
r

where

Basal6.grd5: $prog\_basal = null$
Basal6.grd6: $t \in 0 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$

then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!\!- \{t \mapsto r\}$
end
Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6.delete_setting
any
t
where
Basal6.grd5: $prog\_basal = null$
Basal6.grd2: $basal\_mode \neq suspended$
Basal6.grd6: $t \in 1 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
grd1: $t \neq par\_basal\_update\_rate\_t$
then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!\!- \{t \mapsto -1\}$
end
Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6.add_setting
any
t
r
where
Basal6.grd9: $prog\_basal = null$
Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$
Basal6.grd4: $basal\_mode \neq suspended$
Basal6.grd5: $t \in 0 \mathinner{..} c - 1$
Basal6.grd6: $rate\_setting2(t) = -1$
then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!\!- \{t \mapsto r\}$
end
Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends Basal6.basal_resume_return
when
Basal6.grd8: $prog\_basal = return\_get\_max$
Basal6.grd9: $add\_resume = 2$
then
Basal6.act1: $basal\_rate\_in := max\_value$
Basal6.act2: $basal\_mode := delivering$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $prog\_basal := null$
Basal6.act5: $add\_resume := 0$
act1: $prog := return\_basal\_resume$
act2: $prog1 := return\_basal\_resume$
end
Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.basal_resume_call
when
grd1: $prog = call\_basal\_resume$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in = 0$
Basal6.grd3: ⟨theorem⟩ $basal\_mode = suspended$
Basal6.grd6: $add\_resume = 0$
Basal6.grd5: $prog\_basal = null$
grd2: $prog1 = call\_basal\_resume$
then
Basal6.act1: $par\_get\_t := par\_basal\_resume\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_resume := 1$
end

Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.basal_resume_call_2
when
Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_resume = 1$
then
Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_resume := 2$
end
Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$
extends Basal6·rate_update_return
when
Basal6.grd12: $add\_update = 1$
Basal6.grd4: $prog\_basal = return\_get\_min$
then
Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$
Basal6.act2: $btime := min\_value - par\_get\_t$
Basal6.act3: $add\_update := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_update$
act6: $prog1 := return\_basal\_update$
end
Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.rate_update_call
when
grd1: $prog = call\_basal\_update$
Basal6.grd5: ⟨theorem⟩ $basal\_mode = delivering$
Basal6.grd7: ⟨theorem⟩ $rate\_setting2(par\_basal\_update\_rate\_t) \neq -1$
Basal6.grd3: $add\_update = 0$
Basal6.grd2: $prog\_basal = null$
grd2: $prog1 = call\_basal\_update$
then
Basal6.act1: $par\_get\_t := par\_basal\_update\_rate\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_update := 1$
end
Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_return
when
Basal6.grd8: $add\_start = 2$
Basal6.grd9: $prog\_basal = return\_get\_max$
then
Basal6.act1: $basal\_mode := delivering$
Basal6.act2: $basal\_rate\_in := max\_value$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $add\_start := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_start$
act6: $prog1 := return\_basal\_start$
end
Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call
when
grd1: $prog = call\_basal\_start$
Basal6.grd3: $add\_start = 0$
Basal6.grd4: ⟨theorem⟩ $basal\_mode = stop$
Basal6.grd2: $prog\_basal = null$
grd2: $prog1 = call\_basal\_start$

then
Basal6.act1: $par\_get\_t := par\_basal\_start\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_start := 1$
end
Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call_2
when
Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_start = 1$
then
Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_start := 2$
end
Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$
extends Basal6.stop
when
Basal6.grd2: $prog\_basal = null$
Basal6.grd1: $basal\_mode = delivering$
grd1: $prog = call\_basal\_stop$
grd2: $prog1 = call\_basal\_stop$
then
Basal6.act1: $basal\_mode := stop$
Basal6.act2: $basal\_rate\_in := 0$
act1: $prog := return\_basal\_stop$
act2: $prog1 := return\_basal\_stop$
end
Event Basal6.get_min_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_1
when
Basal6.grd4: $get\_min\_value\_add = 2$
Basal6.grd5: $par\_t = c$
then
Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_2
when
Basal6.grd5: $get\_min\_value\_add = 3$
then
Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_start
when
Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$
then
Basal6.act1: $par\_t := par\_get\_t + 1$
Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$
end
Event Basal6.find_min_value ⟨ordinary⟩ $\widehat{=}$

extends Basal6.find_min_value
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$
then
Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$
end
Event Basal6.find_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value_2
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$
then
Basal6.act1: $temp\_min := par\_t$
Basal6.act2: $get\_min\_value\_add := 3$
end
Event Basal6.get_max_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value
when
Basal6.grd2: $get\_max\_value\_add = 2$
then
Basal6.act3: $prog\_basal := return\_get\_max$
Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$
Basal6.act2: $get\_max\_value\_add := 0$
end
Event Basal6.get_max_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_start
when
Basal6.grd2: $get\_max\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_max$
then
Basal6.act1: $get\_max\_start\_t := par\_get\_t$
Basal6.act2: $get\_max\_value\_add := 1$
Basal6.act3: $par\_t\_max := par\_get\_t$
end
Event Basal6·get_max_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6·get_max_value_1
when
Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd3: $par\_t\_max \geq 0$
Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$
then
Basal6.act1: $par\_t\_max := par\_t\_max - 1$
end
Event Basal6.get_max_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_2
when
Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd2: $par\_t\_max \geq 0$
Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$
then
Basal6.act1: $get\_max\_value\_add := 2$
end
END

MACHINE control_Basal6_NormalBolus_2_Square_Dual_bolus2_3
REFINES control_Basal6_NormalBolus_2_Square_Dual_bolus2
SEES c_normalbolus,c_prog2,c_sd_bolus
VARIABLES

rate_setting2
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
basal_rate_in
basal_mode
btime
par_basal_start_t
par_basal_resume_t
par_basal_update_rate_t
insulin_needed
normal_add
normal_delivering_time
normal_delivering_rate
normal_bolus_suspend
dmodule
d_update_time
state
s_r
s_t
d_deliver_time
d_deliver_rate
d_t
sd_module
sd_flag
prog2
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add
par_t_max
prog_basal

par_get_t

add_resume

add_update

add_start

prog

prog1

INVARIANTS

inv1: $prog2 \in PROG0$

inv2: $prog2 = return\_sd\_resume \Rightarrow d\_deliver\_rate = 0$

inv3: $prog2 = return\_sd\_update \Rightarrow d\_deliver\_rate \in \mathbb{N}_1$

inv4: $prog2 = return\_sd\_suspend \Rightarrow d\_deliver\_rate = 0$

inv5: $prog2 = return\_sd\_resume\_preempt \Rightarrow d\_deliver\_rate \in \mathbb{N}_1$

inv6: $prog2 = return\_sd\_start\_d \Rightarrow d\_deliver\_rate \in \mathbb{N}_1 \land d\_deliver\_time \in \mathbb{N}_1 \land d\_t \in \mathbb{N}_1$

inv7: $prog2 = return\_sd\_start\_d \Rightarrow d\_deliver\_time > d\_t$

inv8: $prog2 = return\_sd\_start\_s \Rightarrow d\_deliver\_rate \in \mathbb{N}_1 \land d\_deliver\_time \in \mathbb{N}_1$

inv15: $prog2 = return\_sd\_finish \Rightarrow d\_deliver\_time = 0$

inv14: $prog2 = return\_sd\_finish \Rightarrow d\_deliver\_rate = 0$

inv16: $prog2 = return\_sd\_start\_s \lor prog2 = call\_sd\_start\_s \Rightarrow sd\_bolus\_work = FALSE \land normal\_bolus\_work = FALSE \land sd\_suspend = FALSE$

inv17: $prog2 = return\_sd\_start\_d \lor prog2 = call\_sd\_start\_d \Rightarrow sd\_bolus\_work = FALSE \land normal\_bolus\_work = FALSE \land sd\_suspend = FALSE$

inv9: $prog2 = return\_sd\_finish \lor prog2 = call\_sd\_finish \Rightarrow sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \land sd\_suspend = FALSE \land time = t\_sd$

inv10: $prog2 = return\_sd\_resume\_preempt \lor prog2 = call\_sd\_resume\_preempt \Rightarrow sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = TRUE \land normal\_bolus\_work = FALSE \land sd\_suspend = FALSE$

inv11: $prog2 = return\_sd\_suspend \lor prog2 = call\_sd\_suspend \Rightarrow sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \land sd\_suspend = FALSE$

inv12: $prog2 = return\_sd\_update \lor prog2 = call\_sd\_update \Rightarrow sd\_suspend = FALSE \land sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \land dmodule = TRUE \land time = d\_update\_time$

inv13: $prog2 = return\_sd\_resume \lor prog2 = call\_sd\_resume \Rightarrow sd\_suspend = TRUE$

inv47: $prog2 = return\_sd\_preempt \lor prog2 = call\_sd\_preempt \Rightarrow sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \land normal\_bolus\_work = FALSE \land sd\_suspend = FALSE$

inv18: $prog2 \in PROG \setminus \{null\} \Rightarrow prog1 = prog2$

inv19: $prog2 = null \Rightarrow prog1 = null$

inv20: $prog2 \in pg2 \Rightarrow prog1 = null$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$

act2: $sd\_bolus\_work := FALSE$

act3: $sd\_preempted\_by\_normal := FALSE$

act7: $sd\_suspend := FALSE$

act8: $normal\_suspend := FALSE$

act9: $basal\_work := FALSE$

act10: $basal\_suspend := FALSE$

act11: $pump\_rate := 0$

act12: $basal\_rate := 0$

act13: $normal\_rate := 0$

act14: $sd\_rate := 0$

act15: $time := 0$

act16: $t\_basal := 0$

act17: $t\_normal := 0$

act18: $t\_sd := 0$

act19: $dmodule := FALSE$

act20: $d\_update\_time := 0$
Basal1.act4: $btime := c$
Basal1.act2: $basal\_rate\_in := 0$
Basal1.act3: $basal\_mode := stop$
Basal6.act15: $prog\_basal := null$
Basal6.act16: $par\_get\_t := 0$
Basal6.act17: $add\_resume := 0$
Basal6.act18: $add\_update := 0$
Basal6.act19: $add\_start := 0$
Basal6.act5: $rate\_setting2 := (1 \mathinner{..} c - 1 \times \{-1\}) \cup \{0 \mapsto 0\}$
Basal6.act6: $min\_value := 0$
Basal6.act7: $max\_value := 0$
Basal6.act11: $get\_min\_value\_add := 0$
Basal6.act8: $par\_t := 0$
Basal6.act9: $temp\_min := 0$
Basal6.act10: $get\_min\_start\_t := 0$
Basal6.act12: $get\_max\_start\_t := 0$
Basal6.act13: $get\_max\_value\_add := 0$
Basal6.act14: $par\_t\_max := 0$
act21: $prog := null$
act22: $par\_basal\_start\_t := 0$
act23: $par\_basal\_resume\_t := 0$
act24: $par\_basal\_update\_rate\_t := 0$
NormalBolus.act1: $insulin\_needed := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_delivering\_rate := 0$
NormalBolus.act4: $normal\_add := 0$
NormalBolus.act5: $normal\_bolus\_suspend := FALSE$
act31: $prog1 := null$
Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act2: $s\_r := 0$
Square_Dual_bolus2.act3: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_module := s$
Square_Dual_bolus2.act10: $sd\_flag := d$
act32: $prog2 := null$

end

Event control5·normal_bolus_start_1_return ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_bolus_start_1_return

any

$t2$

where

grd6: $normal\_bolus\_work = FALSE$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \wedge time \neq d\_update\_time$
control5.grd2: $sd\_bolus\_work = TRUE$
control5.grd3: $sd\_preempted\_by\_normal = FALSE$
control5.grd4: $sd\_suspend = FALSE$
grd1: $prog1 = return\_normal\_start$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$
grd9: $prog2 = return\_normal\_start$

then

control5.act6: $t\_normal := time + normal\_delivering\_time$
control5.act7: $t\_sd := t\_sd - time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $sd\_preempted\_by\_normal := TRUE$
control5.act3: $normal\_rate := normal\_delivering\_rate$
control5.act4: $sd\_rate := 0$

control5.act5: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act8: $d\_update\_time := t2$
act1: $prog1 := null$
act2: $prog2 := null$

end

Event control5·normal_bolus_start_1_call_2 ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_1_call

when

grd1: $prog1 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_preempt$

then

act1: $prog1 := call\_normal\_start$
act2: $prog2 := call\_normal\_start$

end

Event control5·normal_bolus_start_1_call_sd_preempt ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog2 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $dmodule = TRUE \Rightarrow time \neq d\_update\_time$

then

act1: $prog2 := call\_sd\_preempt$

end

Event control5.normal_bolus_start_2_return ⟨ordinary⟩ $\widehat{=}$
extends control5.normal_bolus_start_2_return

when

control5.grd2: $sd\_bolus\_work = FALSE$
grd1: $prog1 = return\_normal\_start$
grd2: $prog2 = return\_normal\_start$

then

control5·act4: $t\_normal := time + normal\_delivering\_time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act1: $prog1 := null$
act2: $prog2 := null$

end

Event control5·normal_bolus_start_2_call ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_2_call

when

grd1: $prog1 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = FALSE$
grd4: $normal\_suspend = FALSE$
grd5: $prog2 = null$

then

act1: $prog1 := call\_normal\_start$
act2: $prog2 := call\_normal\_start$

end

Event control5·normal_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_finish_return

when

grd1: $prog1 = return\_normal\_finish$
grd2: $prog2 = return\_normal\_finish$

then

control5.act4: $t\_normal := 0$
control5.act1: $normal\_bolus\_work := FALSE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
act2: $prog2 := null$

end

Event control5·normal_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$
extends control5 ·normal_bolus_finish_call

when

grd1: $prog1 = null$
grd2: $time = t\_normal$
grd3: $normal\_bolus\_work = TRUE$
grd4: $normal\_suspend = FALSE$
grd5: $prog2 = null$

then

act1: $prog1 := call\_normal\_finish$
act2: $prog2 := call\_normal\_finish$

end

Event control5·normal_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_suspend_return

when

grd1: $prog1 = return\_normal\_suspend$
grd2: $prog2 = return\_normal\_suspend$

then

control5.act4: $t\_normal := t\_normal - time$
control5.act1: $normal\_suspend := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
act3: $prog2 := null$

end

Event control5·normal_bolus_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_suspend_call

when

grd1: $prog1 = null$
grd2: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
grd4: $prog2 = null$

then

act1: $prog1 := call\_normal\_suspend$
act2: $prog2 := call\_normal\_suspend$

end

Event control5·normal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_resume_return

when

grd1: $prog1 = return\_normal\_resume$
grd2: $prog2 = return\_normal\_resume$

then

control5.act4: $t\_normal := 0$
control5.act1: $normal\_suspend := FALSE$
control5.act2: $normal\_rate := 0$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
act2: $normal\_bolus\_work := FALSE$

act3: $prog2 := null$

end

Event control5·normal_bolus_resume_call ⟨ordinary⟩ $\widehat{=}$

extends control5 ·normal_bolus_resume_call

when

grd1: $prog1 = null$

grd2: $normal\_suspend = TRUE$

grd3: $prog2 = null$

then

act1: $prog1 := call\_normal\_resume$

act2: $prog2 := call\_normal\_resume$

end

Event control5·square_or_dual_bolus_start_s_return ⟨ordinary⟩ $\widehat{=}$

refines control5 ·square_or_dual_bolus_start_s

when

grd1: $sd\_bolus\_work = FALSE$

grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_suspend = FALSE$

grd6: $prog2 = return\_sd\_start\_s$

with

r: $r = d\_deliver\_rate$

t: $t = d\_deliver\_time$

then

act1: $sd\_bolus\_work := TRUE$

act2: $sd\_rate := d\_deliver\_rate$

act3: $pump\_rate := d\_deliver\_rate + basal\_rate$

act4: $t\_sd := time + d\_deliver\_time$

act5: $dmodule := FALSE$

act6: $prog2 := null$

end

Event control5square_or_dual_bolus_start_s_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog2 = null$

grd2: $sd\_bolus\_work = FALSE$

grd3: $normal\_bolus\_work = FALSE$

grd4: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_start\_s$

end

Event control5·square_or_dual_bolus_start_d_return ⟨ordinary⟩ $\widehat{=}$

refines control5 ·square_or_dual_bolus_start_d

when

grd1: $sd\_bolus\_work = FALSE$

grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_suspend = FALSE$

grd8: $prog2 = return\_sd\_start\_d$

with

r: $r = d\_deliver\_rate$

t: $t = d\_deliver\_time$

t0: $t0 = d\_t$

then

act1: $sd\_bolus\_work := TRUE$

act5: $dmodule := TRUE$

act6: $d\_update\_time := time + d\_t$

act2: $sd\_rate := d\_deliver\_rate$

act3: $pump\_rate := d\_deliver\_rate + basal\_rate$

act4: $t\_sd := time + d\_deliver\_time$

act7: $prog2 := null$

end

Event control5·square_or_dual_bolus_start_d_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_start\_d$

end

Event control5.square_or_dual_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$

refines control5.square_or_dual_bolus_finish

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$
grd6: $prog2 = return\_sd\_finish$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := d\_deliver\_time$
act5: $dmodule := FALSE$
act6: $prog2 := null$

end

Event control5.square_or_dual_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$

then

act1: $prog2 := call\_sd\_finish$

end

Event control5.square_or_dual_bolus_resume_from_normal_return ⟨ordinary⟩ $\widehat{=}$

refines control5.square_or_dual_bolus_resume_from_normal

any

t2

where

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_resume\_preempt$
grd7: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

with

r: $r = d\_deliver\_rate$

then

act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $prog2 := null$
act6: $d\_update\_time := t2$

end

Event control5.square_or_dual_bolus_resume_from_normal_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = TRUE$
grd4: $normal\_bolus\_work = FALSE$
grd5: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_resume\_preempt$

end

Event control5.sd_suspend_return ⟨ordinary⟩ $\widehat{=}$

refines control5.sd_suspend

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $prog2 = return\_sd\_suspend$

then

act1: $sd\_suspend := TRUE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$
act6: $prog2 := null$
act7: $d\_update\_time := 0$

end

Event control5.sd_suspend_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_suspend$

end

Event control5.square_or_dual_update_rate_return ⟨ordinary⟩ $\widehat{=}$

refines control5.square_or_dual_update_rate

when

grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd6: $dmodule = TRUE$
grd7: $time = d\_update\_time$
grd8: $prog2 = return\_sd\_update$

with

r: $r = d\_deliver\_rate$

then

act1: $sd\_rate := d\_deliver\_rate$
act2: $pump\_rate := d\_deliver\_rate + basal\_rate$
act3: $dmodule := FALSE$
act4: $prog2 := null$
act5: $d\_update\_time := 0$

end

Event control5·square_or_dual_update_rate_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog2 = null$
grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$

grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $dmodule = TRUE$
grd6: $time = d\_update\_time$

then

act1: $prog2 := call\_sd\_update$

end

Event control5.sd_resume_return ⟨ordinary⟩ $\widehat{=}$

refines control5.sd_resume

when

grd1: $sd\_suspend = TRUE$
grd3: $prog2 = return\_sd\_resume$

then

act1: $sd\_suspend := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $prog2 := null$
act6: $sd\_bolus\_work := FALSE$

end

Event control5.sd_resume_call ⟨ordinary⟩ $\widehat{=}$

when

grd1: $prog2 = null$
grd2: $sd\_suspend = TRUE$

then

act1: $prog2 := call\_sd\_resume$

end

Event control5·basal_start_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_return

when

grd1: $prog = return\_basal\_start$
grd2: $prog1 = return\_basal\_start$
grd3: $prog2 = return\_basal\_start$

then

control5.act4: $t\_basal := time + btime$
control5.act1: $basal\_work := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_start_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_call

when

grd1: $basal\_work = FALSE$
grd2: $basal\_suspend = FALSE$
grd3: $prog = null$
grd4: $prog1 = null$
grd5: $prog2 = null$

then

act1: $prog := call\_basal\_start$
act2: $par\_basal\_start\_t := timemodc$
act3: $prog1 := call\_basal\_start$
act4: $prog2 := call\_basal\_start$

end

Event control5·basal_stop_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_stop_return

when

grd1: $prog = return\_basal\_stop$
grd2: $prog1 = return\_basal\_stop$
grd3: $prog2 = return\_basal\_stop$

then

control5.act4: $t\_basal := 0$
control5.act1: $basal\_work := FALSE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_stop_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_stop_call

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd3: $prog = null$
grd4: $prog1 = null$
grd5: $prog2 = null$

then

act1: $prog := call\_basal\_stop$
act2: $prog1 := call\_basal\_stop$
act3: $prog2 := call\_basal\_stop$

end

Event control5·basal_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_suspend_return

when

grd1: $prog = return\_basal\_suspend$
grd2: $prog1 = return\_basal\_suspend$
grd3: $prog2 = return\_basal\_suspend$

then

control5.act4: $t\_basal := 0$
control5.act1: $basal\_suspend := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_suspend_call

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

grd4: $basal\_rate \neq 0$
grd3: $prog = null$
grd5: $prog1 = null$
grd6: $prog2 = null$

then

act1: $prog := call\_basal\_suspend$
act2: $prog1 := call\_basal\_suspend$
act3: $prog2 := call\_basal\_suspend$

end

Event control5·basal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_resume_return

when

grd1: $prog = return\_basal\_resume$
grd2: $prog1 = return\_basal\_resume$
grd3: $prog2 = return\_basal\_resume$

then

control5.act1: $basal\_suspend := FALSE$
control5.act4: $t\_basal := btime + time$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_resume_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_resume_call

when

grd1: $basal\_suspend = TRUE$
grd2: $prog = null$
grd3: $prog1 = null$
grd4: $prog2 = null$

then

act1: $prog := call\_basal\_resume$
act2: $par\_basal\_resume\_t := timemodc$
act3: $prog1 := call\_basal\_resume$
act4: $prog2 := call\_basal\_resume$

end

Event control5·basal_update_rate_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·basal_update_rate_return

when

grd1: $prog = return\_basal\_update$
grd2: $prog1 = return\_basal\_update$
grd3: $prog2 = return\_basal\_update$

then

control5.act3: $t\_basal := time + btime$
control5.act1: $basal\_rate := basal\_rate\_in$
control5.act2: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_update_rate_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_update_rate_call

when

grd1: $t\_basal = time$
grd2: $basal\_suspend = FALSE$
grd3: $basal\_work = TRUE$
grd4: $prog = null$
grd5: $t\_basal \in dom(rate\_setting2 \rhd \{-1\})$
grd6: $prog1 = null$
grd7: $prog2 = null$

then

act1: $prog := call\_basal\_update$
act2: $par\_basal\_update\_rate\_t := t\_basal$
act3: $prog1 := call\_basal\_update$
act4: $prog2 := call\_basal\_update$

end

Event control5.timer ⟨ordinary⟩ $\widehat{=}$
extends control5.timer

when

grd1:
¬(
((normal_bolus_work = TRUE) ∧ (normal_suspend = FALSE) ∧ (time = t_normal)) ∨
((sd_bolus_work = TRUE) ∧ (sd_preempted_by_normal = FALSE) ∧ (sd_suspend = FALSE) ∧
(time = t_sd)) ∨
(((sd_suspend = FALSE) ∧ (sd_bolus_work = TRUE) ∧ (sd_preempted_by_normal = FALSE) ∧
(dmodule = TRUE) ∧ (time = d_update_time))) ∨
(((basal_suspend = FALSE) ∧ (basal_work = TRUE) ∧ (t_basal = time)))
)

then

act1: $time := time + 1$

end

Event NormalBolus.normal_bolus_start_calculate_insulin_needed ⟨ordinary⟩ ≙

extends NormalBolus.normal_bolus_start_calculate_insulin_needed

any

$insulin$

where

NormalBolus.grd1: $insulin > 0$

NormalBolus.grd3: $normal\_add = 0$

grd1: $prog1 = call\_normal\_start$

grd2: $prog2 = call\_normal\_start$

then

NormalBolus.act1: $insulin\_needed := insulin$

NormalBolus.act2: $normal\_add := 1$

end

Event NormalBolus.normal_bolus_start_calculate_lasting_time ⟨ordinary⟩ ≙

extends NormalBolus.normal_bolus_start_calculate_lasting_time

when

NormalBolus.grd1: $normal\_add = 1$

then

NormalBolus.act1: $normal\_delivering\_time := insulin\_needed / normal\_bolus\_rate$

NormalBolus.act2: $insulin\_needed := 0$

NormalBolus.act3: $normal\_add := 2$

end

Event NormalBolus.normal_bolus_delivery ⟨ordinary⟩ ≙

extends NormalBolus.normal_bolus_delivery

when

NormalBolus.grd2: $normal\_add = 2$

then

NormalBolus.act1: $normal\_delivering\_rate := normal\_bolus\_rate$

NormalBolus.act2: $normal\_add := 3$

act1: $prog1 := return\_normal\_start$

act2: $prog2 := return\_normal\_start$

end

Event NormalBolus.normal_bolus_suspend ⟨ordinary⟩ ≙

extends NormalBolus.normal_bolus_suspend

when

grd1: $prog1 = call\_normal\_suspend$

grd2: $prog2 = call\_normal\_suspend$

then

NormalBolus.act1: $normal\_delivering\_rate := 0$

NormalBolus.act2: $normal\_delivering\_time := 0$

NormalBolus.act3: $normal\_add := 0$

NormalBolus.act4: $normal\_bolus\_suspend := TRUE$

act1: $prog1 := return\_normal\_suspend$

act2: $prog2 := return\_normal\_suspend$

end

Event NormalBolus.normal_bolus_finish ⟨ordinary⟩ ≙

extends NormalBolus.normal_bolus_finish

when

grd1: $prog1 = call\_normal\_finish$

grd2: $prog2 = call\_normal\_finish$

then

NormalBolus.act1: $normal\_delivering\_rate := 0$

NormalBolus.act2: $normal\_delivering\_time := 0$

NormalBolus.act3: $normal\_add := 0$

act1: $prog1 := return\_normal\_finish$

act2: $prog2 := return\_normal\_finish$

end

Event NormalBolus.normal_bolus_resume ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_resume

when

grd1: $prog1 = call\_normal\_resume$

grd2: $prog2 = call\_normal\_resume$

then

NormalBolus.act1: $normal\_bolus\_suspend := FALSE$

act1: $normal\_delivering\_rate := 0$

act2: $prog1 := return\_normal\_resume$

act3: $prog2 := return\_normal\_resume$

end

Event Square_Dual_bolus2.start ⟨ordinary⟩ $\widehat{=}$

extends Square_Dual_bolus2.start

any

t

r

where

Square_Dual_bolus2.grd1: $state = off$

Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$

Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$

grd2: $prog2 = call\_sd\_start\_s$

then

Square_Dual_bolus2.act1: $state := deliver$

Square_Dual_bolus2.act2: $s\_r := r$

Square_Dual_bolus2.act3: $s\_t := t$

Square_Dual_bolus2.act6: $sd\_module := s$

Square_Dual_bolus2.act7: $d\_deliver\_time := t$

Square_Dual_bolus2.act8: $d\_deliver\_rate := r$

act2: $prog2 := return\_sd\_start\_s$

end

Event Square_Dual_bolus2.start_dual ⟨ordinary⟩ $\widehat{=}$

extends Square_Dual_bolus2.start_dual

any

t

r

td

where

Square_Dual_bolus2.grd1: $state = off$

Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$

Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$

Square_Dual_bolus2.grd4: $td \in \mathbb{N}_1$

grd2: $prog2 = call\_sd\_start\_d$

then

Square_Dual_bolus2.act1: $state := deliver$

Square_Dual_bolus2.act2: $s\_r := r$

Square_Dual_bolus2.act3: $s\_t := t$

Square_Dual_bolus2.act6: $d\_deliver\_time := t + td$

Square_Dual_bolus2.act7: $d\_deliver\_rate := normal\_bolus\_rate$
Square_Dual_bolus2.act8: $d\_t := td$
Square_Dual_bolus2.act9: $sd\_module := d$
act2: $prog2 := return\_sd\_start\_d$

end

Event Square_Dual_bolus2.update_to_dual ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.update_to_dual

when

Square_Dual_bolus2.grd2: $state = deliver$
Square_Dual_bolus2.grd3: $sd\_module = d$
Square_Dual_bolus2.grd4: $sd\_flag = d$
grd2: $prog2 = call\_sd\_update$

then

Square_Dual_bolus2.act2: $d\_deliver\_rate := s\_r$
Square_Dual_bolus2.act3: $sd\_flag := s$
act2: $prog2 := return\_sd\_update$

end

Event Square_Dual_bolus2.finish ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.finish

when

Square_Dual_bolus2.grd1: $state = deliver$
Square_Dual_bolus2.grd2: $sd\_module = d \Rightarrow sd\_flag = s$
grd1: $prog2 = call\_sd\_finish$

then

Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act4: $s\_r := 0$
Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_finish$

end

Event Square_Dual_bolus2.suspend ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.suspend

when

Square_Dual_bolus2.grd1: $state = deliver$
grd1: $prog2 = call\_sd\_suspend$

then

Square_Dual_bolus2.act1: $state := suspend$
Square_Dual_bolus2.act4: $s\_r := 0$
Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_suspend$

end

Event Square_Dual_bolus2.resume ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.resume

when

Square_Dual_bolus2.grd1: $state = suspend$
grd1: $prog2 = call\_sd\_resume$

then

Square_Dual_bolus2.act1: $state := off$
act1: $prog2 := return\_sd\_resume$

end

Event Square_Dual_bolus2.preempted ⟨ordinary⟩ $\widehat{=}$

extends Square_Dual_bolus2.preempted

any

t time left for square bolus

where

Square_Dual_bolus2.grd1: $state = deliver$

Square_Dual_bolus2.grd2: $t \in 0 \mathinner{..} d\_deliver\_time$

grd1: $prog2 = call\_sd\_preempt$

then

Square_Dual_bolus2.act1: $state := preempt$

Square_Dual_bolus2.act4: $d\_deliver\_time := t$

Square_Dual_bolus2.act5: $d\_deliver\_rate := 0$

act1: $prog2 := return\_sd\_preempt$

end

Event Square_Dual_bolus2.resume_from_preempt ⟨ordinary⟩ $\widehat{=}$

extends Square_Dual_bolus2.resume_from_preempt

any

r

where

Square_Dual_bolus2.grd1: $state = preempt$

Square_Dual_bolus2.grd2: $sd\_module = s \Rightarrow r = s\_r$

Square_Dual_bolus2.grd3: $sd\_module = d \wedge sd\_flag = d \Rightarrow r = normal\_bolus\_rate$

Square_Dual_bolus2.grd4: $sd\_module = d \wedge sd\_flag = s \Rightarrow r = s\_r$

grd1: $prog2 = call\_sd\_resume\_preempt$

then

Square_Dual_bolus2.act1: $state := deliver$

Square_Dual_bolus2.act4: $d\_deliver\_rate := r$

act1: $prog2 := return\_sd\_resume\_preempt$

end

Event Basal6·basal_suspend ⟨ordinary⟩ $\widehat{=}$

extends Basal6·basal_suspend

when

grd1: $prog = call\_basal\_suspend$

Basal6.grd3: $prog\_basal = null$

Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in \neq 0$

Basal6.grd2: ⟨theorem⟩ $basal\_mode = delivering$

grd2: $prog1 = call\_basal\_suspend$

grd3: $prog2 = call\_basal\_suspend$

then

Basal6.act1: $basal\_rate\_in := 0$

Basal6.act2: $basal\_mode := suspended$

act1: $prog := return\_basal\_suspend$

act2: $prog1 := return\_basal\_suspend$

act3: $prog2 := return\_basal\_suspend$

end

Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6·change_setting

any

t

r

where

Basal6.grd5: $prog\_basal = null$

Basal6.grd6: $t \in 0 \mathinner{..} c - 1$

Basal6.grd7: $rate\_setting2(t) \neq -1$

Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \Leftarrow \{t \mapsto r\}$

end

Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.delete_setting
any
t
where
Basal6.grd5: $prog\_basal = null$
Basal6.grd2: $basal\_mode \neq suspended$
Basal6.grd6: $t \in 1 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
grd1: $t \neq par\_basal\_update\_rate\_t$
then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!\!- \{t \mapsto -1\}$
end
Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6.add_setting
any
t
r
where
Basal6.grd9: $prog\_basal = null$
Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$
Basal6.grd4: $basal\_mode \neq suspended$
Basal6.grd5: $t \in 0 \mathinner{..} c - 1$
Basal6.grd6: $rate\_setting2(t) = -1$
then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!\!- \{t \mapsto r\}$
end
Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends Basal6.basal_resume_return
when
Basal6.grd8: $prog\_basal = return\_get\_max$
Basal6.grd9: $add\_resume = 2$
then
Basal6.act1: $basal\_rate\_in := max\_value$
Basal6.act2: $basal\_mode := delivering$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $prog\_basal := null$
Basal6.act5: $add\_resume := 0$
act1: $prog := return\_basal\_resume$
act2: $prog1 := return\_basal\_resume$
act3: $prog2 := return\_basal\_resume$
end
Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.basal_resume_call
when
grd1: $prog = call\_basal\_resume$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in = 0$
Basal6.grd3: ⟨theorem⟩ $basal\_mode = suspended$
Basal6.grd6: $add\_resume = 0$
Basal6.grd5: $prog\_basal = null$
grd2: $prog1 = call\_basal\_resume$
grd3: $prog2 = call\_basal\_resume$
then
Basal6.act1: $par\_get\_t := par\_basal\_resume\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_resume := 1$
end
Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.basal_resume_call_2

when
Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_resume = 1$
then
Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_resume := 2$
end
Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$
extends Basal6·rate_update_return
when
Basal6.grd12: $add\_update = 1$
Basal6.grd4: $prog\_basal = return\_get\_min$
then
Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$
Basal6.act2: $btime := min\_value - par\_get\_t$
Basal6.act3: $add\_update := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_update$
act6: $prog1 := return\_basal\_update$
act7: $prog2 := return\_basal\_update$
end
Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.rate_update_call
when
grd1: $prog = call\_basal\_update$
Basal6.grd5: ⟨theorem⟩ $basal\_mode = delivering$
Basal6.grd7: ⟨theorem⟩ $rate\_setting2(par\_basal\_update\_rate\_t) \neq -1$
Basal6.grd3: $add\_update = 0$
Basal6.grd2: $prog\_basal = null$
grd2: $prog1 = call\_basal\_update$
grd3: $prog2 = call\_basal\_update$
then
Basal6.act1: $par\_get\_t := par\_basal\_update\_rate\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_update := 1$
end
Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_return
when
Basal6.grd8: $add\_start = 2$
Basal6.grd9: $prog\_basal = return\_get\_max$
then
Basal6.act1: $basal\_mode := delivering$
Basal6.act2: $basal\_rate\_in := max\_value$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $add\_start := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_start$
act6: $prog1 := return\_basal\_start$
act7: $prog2 := return\_basal\_start$
end
Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call
when
grd1: $prog = call\_basal\_start$
Basal6.grd3: $add\_start = 0$
Basal6.grd4: ⟨theorem⟩ $basal\_mode = stop$
Basal6.grd2: $prog\_basal = null$

grd2: $prog1 = call\_basal\_start$
grd3: $prog2 = call\_basal\_start$
then
Basal6.act1: $par\_get\_t := par\_basal\_start\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_start := 1$
end
Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call_2
when
Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_start = 1$
then
Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_start := 2$
end
Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$
extends Basal6.stop
when
Basal6.grd2: $prog\_basal = null$
Basal6.grd1: $basal\_mode = delivering$
grd1: $prog = call\_basal\_stop$
grd2: $prog1 = call\_basal\_stop$
grd3: $prog2 = call\_basal\_stop$
then
Basal6.act1: $basal\_mode := stop$
Basal6.act2: $basal\_rate\_in := 0$
act1: $prog := return\_basal\_stop$
act2: $prog1 := return\_basal\_stop$
act3: $prog2 := return\_basal\_stop$
end
Event Basal6.get_min_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_1
when
Basal6.grd4: $get\_min\_value\_add = 2$
Basal6.grd5: $par\_t = c$
then
Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_2
when
Basal6.grd5: $get\_min\_value\_add = 3$
then
Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_start
when
Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$
then
Basal6.act1: $par\_t := par\_get\_t + 1$

Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$

end

Event Basal6.find_min_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value

when

Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$

then

Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$

end

Event Basal6.find_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value_2

when

Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$

then

Basal6.act1: $temp\_min := par\_t$
Basal6.act2: $get\_min\_value\_add := 3$

end

Event Basal6.get_max_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value

when

Basal6.grd2: $get\_max\_value\_add = 2$

then

Basal6.act3: $prog\_basal := return\_get\_max$
Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$
Basal6.act2: $get\_max\_value\_add := 0$

end

Event Basal6.get_max_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_start

when

Basal6.grd2: $get\_max\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_max$

then

Basal6.act1: $get\_max\_start\_t := par\_get\_t$
Basal6.act2: $get\_max\_value\_add := 1$
Basal6.act3: $par\_t\_max := par\_get\_t$

end

Event Basal6·get_max_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6·get_max_value_1

when

Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd3: $par\_t\_max \geq 0$
Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$

then

Basal6.act1: $par\_t\_max := par\_t\_max - 1$

end

Event Basal6.get_max_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_2

when

Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd2: $par\_t\_max \geq 0$
Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$

then

Basal6.act1: $get\_max\_value\_add := 2$

end

END

MACHINE control_Basal6_NormalBolus_2_Square_Dual_bolus2_4
REFINES control_Basal6_NormalBolus_2_Square_Dual_bolus2_3
SEES c_normalbolus_anim,c_prog2_anim,c_basal_anim,c_sd_bolus
VARIABLES

rate_setting2
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
basal_rate_in
basal_mode
btime
par_basal_start_t
par_basal_resume_t
par_basal_update_rate_t
insulin_needed
normal_add
normal_delivering_time
normal_delivering_rate
normal_bolus_suspend
dmodule
d_update_time
state
s_r
s_t
d_deliver_time
d_deliver_rate
d_t
sd_module
sd_flag
prog2
par_sd_preempt_t
sdp_add
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add

par_t_max

prog_basal

par_get_t

add_resume

add_update

add_start

prog

prog1

INVARIANTS

inv22: $prog2 = call\_sd\_start\_s \Rightarrow state = off$

inv23: $prog2 = call\_sd\_start\_d \Rightarrow state = off$

inv24: $prog2 = call\_sd\_update \Rightarrow state = deliver \land sd\_module = d \land sd\_flag = d$

inv25: $prog2 = call\_sd\_finish \Rightarrow state = deliver \land (sd\_module = d \Rightarrow sd\_flag = s)$

inv26: $prog2 = call\_sd\_suspend \Rightarrow state = deliver$

inv28: $prog2 = call\_sd\_preempt \Rightarrow state = deliver$

inv32: $prog2 = return\_sd\_resume \lor prog2 = return\_sd\_finish \Rightarrow state = off$

inv33: $prog2 \in PROG \land sd\_bolus\_work = FALSE \Rightarrow state = off$

inv35: $prog2 \in PROG \setminus \{call\_normal\_start, return\_normal\_start\} \land sd\_suspend = FALSE \land sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \Rightarrow state = deliver$

inv39: $normal\_add \in \{1, 2\} \Rightarrow prog2 = call\_normal\_start$

inv38: $prog2 = return\_sd\_preempt \Rightarrow state = preempt$

inv37: $prog2 \in PROG \land dmodule = TRUE \land time \leq d\_update\_time \land sd\_suspend = FALSE \land sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \Rightarrow sd\_module = d \land sd\_flag = d$

inv42: $prog2 \in PROG \cup \{call\_sd\_resume\_preempt, return\_sd\_resume\_preempt, return\_sd\_preempt, call\_sd\_preempt\} \land dmodule = TRUE \Rightarrow sd\_module = d \land sd\_flag = d$

inv43: $prog2 = return\_sd\_start\_d \Rightarrow sd\_module = d \land sd\_flag = d$

inv40: $prog2 \in PROG \land sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \land sd\_suspend = FALSE \land time = t\_sd \Rightarrow dmodule = FALSE$
(sd_module=d ⇒ sd_flag=s)

inv41: $prog2 \in PROG \cup \{call\_sd\_resume\_preempt, return\_sd\_resume\_preempt, return\_sd\_preempt, call\_sd\_preempt, call\_normal\_start, return\_normal\_start\} \land sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \land sd\_suspend = FALSE \land dmodule = FALSE \Rightarrow (sd\_module = d \Rightarrow sd\_flag = s)$

inv44: $prog2 \in \{return\_sd\_update, return\_sd\_start\_s\} \Rightarrow (sd\_module = d \Rightarrow sd\_flag = s)$

inv45: $prog2 = return\_sd\_finish \Rightarrow sd\_flag = d$

inv46: $sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = TRUE \land sd\_suspend = FALSE \land dmodule = FALSE \Rightarrow (sd\_module = d \Rightarrow sd\_flag = s)$

inv1: $prog2 \in PROG \land sd\_suspend = TRUE \Rightarrow state = suspend$

inv2: $prog2 \in PROG \land sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = TRUE \land sd\_suspend = FALSE \Rightarrow state = preempt$

inv3: $prog2 = return\_sd\_suspend \Rightarrow state = suspend$

inv4: $prog2 \in \{call\_normal\_start, return\_normal\_start\} \land sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \land normal\_bolus\_work = FALSE \land sd\_suspend = FALSE \Rightarrow state = preempt$

inv5: $prog2 = return\_sd\_preempt \Rightarrow state = preempt$

inv6: $prog2 = call\_sd\_resume \Rightarrow state = suspend$

inv7: $prog2 = call\_sd\_resume\_preempt \Rightarrow state = preempt$

inv47: $par\_sd\_preempt\_t \in \mathbb{N}$

inv48: $prog2 = call\_sd\_preempt \Rightarrow par\_sd\_preempt\_t \in 0 \mathinner{..} d\_deliver\_time$

inv49: $prog2 = call\_sd\_preempt \Rightarrow t\_sd - time \geq 0$

inv51: $prog2 \in PROG \cup \{call\_sd\_preempt, call\_sd\_update, return\_sd\_update\} \land normal\_bolus\_work = FALSE \land sd\_bolus\_work = TRUE \land sd\_preempted\_by\_normal = FALSE \land sd\_suspend = FALSE \Rightarrow t\_sd - time \leq d\_deliver\_time$

inv53: $prog2 \in \{return\_sd\_start\_s, return\_sd\_start\_d\} \land sd\_bolus\_work = FALSE \land sd\_suspend = FALSE \Rightarrow t\_sd - time \leq d\_deliver\_time$

inv54: $prog2 \in \{call\_sd\_resume\_preempt, return\_sd\_resume\_preempt\} \Rightarrow t\_sd \leq d\_deliver\_time$

inv55: $sd\_preempted\_by\_normal = TRUE \Rightarrow t\_sd \leq d\_deliver\_time$

inv56: $prog2 \in \{return\_sd\_preempt\} \Rightarrow d\_deliver\_time = par\_sd\_preempt\_t \land par\_sd\_preempt\_t = t\_sd - time$

inv57: $prog2 \in \{call\_normal\_start, return\_normal\_start\} \land sd\_bolus\_work = TRUE \Rightarrow d\_deliver\_time = par\_sd\_preempt\_t \land par\_sd\_preempt\_t = t\_sd - time$

inv58: $prog2 = call\_sd\_preempt \Rightarrow par\_sd\_preempt\_t = t\_sd - time$

inv59: $sdp\_add \in 0 \,..\, 1$

inv60: $prog2 \in \{return\_sd\_preempt, call\_sd\_preempt\} \lor (prog2 \in \{call\_normal\_start, return\_normal\_start\} \land sd\_bolus\_work = TRUE) \Rightarrow sdp\_add = 1$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$

act2: $sd\_bolus\_work := FALSE$

act3: $sd\_preempted\_by\_normal := FALSE$

act7: $sd\_suspend := FALSE$

act8: $normal\_suspend := FALSE$

act9: $basal\_work := FALSE$

act10: $basal\_suspend := FALSE$

act11: $pump\_rate := 0$

act12: $basal\_rate := 0$

act13: $normal\_rate := 0$

act14: $sd\_rate := 0$

act15: $time := 0$

act16: $t\_basal := 0$

act17: $t\_normal := 0$

act18: $t\_sd := 0$

act19: $dmodule := FALSE$

act20: $d\_update\_time := 0$

Basal1.act4: $btime := c$

Basal1.act2: $basal\_rate\_in := 0$

Basal1.act3: $basal\_mode := stop$

Basal6.act15: $prog\_basal := null$

Basal6.act16: $par\_get\_t := 0$

Basal6.act17: $add\_resume := 0$

Basal6.act18: $add\_update := 0$

Basal6.act19: $add\_start := 0$

Basal6.act5: $rate\_setting2 := (1 \,..\, c - 1 \times \{-1\}) \cup \{0 \mapsto 0\}$

Basal6.act6: $min\_value := 0$

Basal6.act7: $max\_value := 0$

Basal6.act11: $get\_min\_value\_add := 0$

Basal6.act8: $par\_t := 0$

Basal6.act9: $temp\_min := 0$

Basal6.act10: $get\_min\_start\_t := 0$

Basal6.act12: $get\_max\_start\_t := 0$

Basal6.act13: $get\_max\_value\_add := 0$

Basal6.act14: $par\_t\_max := 0$

act21: $prog := null$

act22: $par\_basal\_start\_t := 0$

act23: $par\_basal\_resume\_t := 0$

act24: $par\_basal\_update\_rate\_t := 0$

NormalBolus.act1: $insulin\_needed := 0$

NormalBolus.act2: $normal\_delivering\_time := 0$

NormalBolus.act3: $normal\_delivering\_rate := 0$

NormalBolus.act4: $normal\_add := 0$

NormalBolus.act5: $normal\_bolus\_suspend := FALSE$

act31: $prog1 := null$

Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act2: $s\_r := 0$
Square_Dual_bolus2.act3: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_module := s$
Square_Dual_bolus2.act10: $sd\_flag := d$
act32: $prog2 := null$
act33: $par\_sd\_preempt\_t := 0$
act34: $sdp\_add := 0$

end

Event control5·normal_bolus_start_1_return ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_bolus_start_1_return

any

$t2$

where

grd6: $normal\_bolus\_work = FALSE$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \land time \neq d\_update\_time$
control5.grd2: $sd\_bolus\_work = TRUE$
control5.grd3: $sd\_preempted\_by\_normal = FALSE$
control5.grd4: $sd\_suspend = FALSE$
grd1: $prog1 = return\_normal\_start$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$
grd9: $prog2 = return\_normal\_start$
grd4: ⟨theorem⟩ $sdp\_add = 1$

then

control5.act6: $t\_normal := time + normal\_delivering\_time$
control5.act7: $t\_sd := t\_sd - time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $sd\_preempted\_by\_normal := TRUE$
control5.act3: $normal\_rate := normal\_delivering\_rate$
control5.act4: $sd\_rate := 0$
control5.act5: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act8: $d\_update\_time := t2$
act1: $prog1 := null$
act2: $prog2 := null$
act3: $sdp\_add := 0$

end

Event control5·normal_bolus_start_1_call_2 ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_bolus_start_1_call_2

when

grd1: $prog1 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_preempt$
grd8: ⟨theorem⟩ $sdp\_add = 1$

then

act1: $prog1 := call\_normal\_start$
act2: $prog2 := call\_normal\_start$

end

Event control5·normal_bolus_start_1_call_sd_preempt ⟨ordinary⟩ $\widehat{=}$

extends control5 ·normal_bolus_start_1_call_sd_preempt

when

grd1: $prog2 = null$
grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $dmodule = TRUE \Rightarrow time \neq d\_update\_time$
grd7: $sdp\_add = 0$

then

act1: $prog2 := call\_sd\_preempt$
act2: $par\_sd\_preempt\_t := t\_sd - time$
act3: $sdp\_add := 1$

end

Event control5.normal_bolus_start_2_return ⟨ordinary⟩ $\widehat{=}$

extends control5.normal_bolus_start_2_return

when

control5.grd2: $sd\_bolus\_work = FALSE$
grd1: $prog1 = return\_normal\_start$
grd2: $prog2 = return\_normal\_start$

then

control5·act4: $t\_normal := time + normal\_delivering\_time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act1: $prog1 := null$
act2: $prog2 := null$

end

Event control5·normal_bolus_start_2_call ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_bolus_start_2_call

when

grd1: $prog1 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = FALSE$
grd4: $normal\_suspend = FALSE$
grd5: $prog2 = null$

then

act1: $prog1 := call\_normal\_start$
act2: $prog2 := call\_normal\_start$

end

Event control5·normal_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_bolus_finish_return

when

grd1: $prog1 = return\_normal\_finish$
grd2: $prog2 = return\_normal\_finish$

then

control5.act4: $t\_normal := 0$
control5.act1: $normal\_bolus\_work := FALSE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
act2: $prog2 := null$

end

Event control5·normal_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$

extends control5 ·normal_bolus_finish_call

when

grd1: $prog1 = null$
grd2: $time = t\_normal$
grd3: $normal\_bolus\_work = TRUE$
grd4: $normal\_suspend = FALSE$
grd5: $prog2 = null$

then

act1: $prog1 := call\_normal\_finish$
act2: $prog2 := call\_normal\_finish$

end

Event control5·normal_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_suspend_return

when

grd1: $prog1 = return\_normal\_suspend$
grd2: $prog2 = return\_normal\_suspend$

then

control5.act4: $t\_normal := t\_normal - time$
control5.act1: $normal\_suspend := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
act3: $prog2 := null$

end

Event control5·normal_bolus_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_suspend_call

when

grd1: $prog1 = null$
grd2: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
grd4: $prog2 = null$

then

act1: $prog1 := call\_normal\_suspend$
act2: $prog2 := call\_normal\_suspend$

end

Event control5·normal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_resume_return

when

grd1: $prog1 = return\_normal\_resume$
grd2: $prog2 = return\_normal\_resume$

then

control5.act4: $t\_normal := 0$
control5.act1: $normal\_suspend := FALSE$
control5.act2: $normal\_rate := 0$
control5.act3: $pump\_rate := basal\_rate$
act1: $prog1 := null$
act2: $normal\_bolus\_work := FALSE$
act3: $prog2 := null$

end

Event control5·normal_bolus_resume_call ⟨ordinary⟩ $\widehat{=}$
extends control5 ·normal_bolus_resume_call

when

grd1: $prog1 = null$
grd2: $normal\_suspend = TRUE$
grd3: $prog2 = null$

then

act1: $prog1 := call\_normal\_resume$
act2: $prog2 := call\_normal\_resume$

end

Event control5·square_or_dual_bolus_start_s_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·square_or_dual_bolus_start_s_return

when

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$

grd6: $prog2 = return\_sd\_start\_s$

then

act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + d\_deliver\_time$
act5: $dmodule := FALSE$
act6: $prog2 := null$

end

Event control5square_or_dual_bolus_start_s_call ⟨ordinary⟩ $\widehat{=}$
extends control5square_or_dual_bolus_start_s_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_start\_s$

end

Event control5·square_or_dual_bolus_start_d_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·square_or_dual_bolus_start_d_return

when

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd8: $prog2 = return\_sd\_start\_d$

then

act1: $sd\_bolus\_work := TRUE$
act5: $dmodule := TRUE$
act6: $d\_update\_time := time + d\_t$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + d\_deliver\_time$
act7: $prog2 := null$

end

Event control5·square_or_dual_bolus_start_d_call ⟨ordinary⟩ $\widehat{=}$
extends control5·square_or_dual_bolus_start_d_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_start\_d$

end

Event control5.square_or_dual_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_finish_return

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$
grd6: $prog2 = return\_sd\_finish$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := d\_deliver\_time$

act5: $dmodule := FALSE$
act6: $prog2 := null$

end

Event control5.square_or_dual_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_finish_call

when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$

then
act1: $prog2 := call\_sd\_finish$

end

Event control5.square_or_dual_bolus_resume_from_normal_return ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_resume_from_normal_return

any
$t2$

where
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_resume\_preempt$
grd7: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

then
act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $prog2 := null$
act6: $d\_update\_time := t2$

end

Event control5.square_or_dual_bolus_resume_from_normal_call ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_resume_from_normal_call

when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = TRUE$
grd4: $normal\_bolus\_work = FALSE$
grd5: $sd\_suspend = FALSE$

then
act1: $prog2 := call\_sd\_resume\_preempt$

end

Event control5.sd_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_suspend_return

when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $prog2 = return\_sd\_suspend$

then
act1: $sd\_suspend := TRUE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$

act6: $prog2 := null$
act7: $d\_update\_time := 0$

end

Event control5.sd_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_suspend_call
when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $prog2 := call\_sd\_suspend$
end

Event control5.square_or_dual_update_rate_return ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_update_rate_return
when
grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd6: $dmodule = TRUE$
grd7: $time = d\_update\_time$
grd8: $prog2 = return\_sd\_update$
then
act1: $sd\_rate := d\_deliver\_rate$
act2: $pump\_rate := d\_deliver\_rate + basal\_rate$
act3: $dmodule := FALSE$
act4: $prog2 := null$
act5: $d\_update\_time := 0$
end

Event control5·square_or_dual_update_rate_call ⟨ordinary⟩ $\widehat{=}$
extends control5·square_or_dual_update_rate_call
when
grd1: $prog2 = null$
grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $dmodule = TRUE$
grd6: $time = d\_update\_time$
then
act1: $prog2 := call\_sd\_update$
end

Event control5.sd_resume_return ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_resume_return
when
grd1: $sd\_suspend = TRUE$
grd3: $prog2 = return\_sd\_resume$
then
act1: $sd\_suspend := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $prog2 := null$
act6: $sd\_bolus\_work := FALSE$
end

Event control5.sd_resume_call ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_resume_call
when

grd1: $prog2 = null$
grd2: $sd\_suspend = TRUE$

then

act1: $prog2 := call\_sd\_resume$

end

Event control5·basal_start_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_return

when

grd1: $prog = return\_basal\_start$
grd2: $prog1 = return\_basal\_start$
grd3: $prog2 = return\_basal\_start$

then

control5.act4: $t\_basal := time + btime$
control5.act1: $basal\_work := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_start_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_call

when

grd1: $basal\_work = FALSE$
grd2: $basal\_suspend = FALSE$
grd3: $prog = null$
grd4: $prog1 = null$
grd5: $prog2 = null$

then

act1: $prog := call\_basal\_start$
act2: $par\_basal\_start\_t := timemodc$
act3: $prog1 := call\_basal\_start$
act4: $prog2 := call\_basal\_start$

end

Event control5·basal_stop_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_stop_return

when

grd1: $prog = return\_basal\_stop$
grd2: $prog1 = return\_basal\_stop$
grd3: $prog2 = return\_basal\_stop$

then

control5.act4: $t\_basal := 0$
control5.act1: $basal\_work := FALSE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_stop_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_stop_call

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd3: $prog = null$
grd4: $prog1 = null$
grd5: $prog2 = null$

then

act1: $prog := call\_basal\_stop$
act2: $prog1 := call\_basal\_stop$
act3: $prog2 := call\_basal\_stop$

end

Event control5·basal_suspend_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_suspend_return

when

grd1: $prog = return\_basal\_suspend$
grd2: $prog1 = return\_basal\_suspend$
grd3: $prog2 = return\_basal\_suspend$

then

control5.act4: $t\_basal := 0$
control5.act1: $basal\_suspend := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_suspend_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_suspend_call

when

grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

grd4: $basal\_rate \neq 0$
grd3: $prog = null$
grd5: $prog1 = null$
grd6: $prog2 = null$

then

act1: $prog := call\_basal\_suspend$
act2: $prog1 := call\_basal\_suspend$
act3: $prog2 := call\_basal\_suspend$

end

Event control5·basal_resume_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_resume_return

when

grd1: $prog = return\_basal\_resume$
grd2: $prog1 = return\_basal\_resume$
grd3: $prog2 = return\_basal\_resume$

then

control5.act1: $basal\_suspend := FALSE$
control5.act4: $t\_basal := btime + time$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_resume_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_resume_call

when

grd1: $basal\_suspend = TRUE$
grd2: $prog = null$
grd3: $prog1 = null$
grd4: $prog2 = null$

then

act1: $prog := call\_basal\_resume$

act2: $par\_basal\_resume\_t := timemodc$
act3: $prog1 := call\_basal\_resume$
act4: $prog2 := call\_basal\_resume$

end

Event control5·basal_update_rate_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·basal_update_rate_return

when

grd1: $prog = return\_basal\_update$
grd2: $prog1 = return\_basal\_update$
grd3: $prog2 = return\_basal\_update$

then

control5.act3: $t\_basal := time + btime$
control5.act1: $basal\_rate := basal\_rate\_in$
control5.act2: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act1: $prog := null$
act2: $prog1 := null$
act3: $prog2 := null$

end

Event control5·basal_update_rate_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_update_rate_call

when

grd1: $t\_basal = time$
grd2: $basal\_suspend = FALSE$
grd3: $basal\_work = TRUE$
grd4: $prog = null$
grd5: $t\_basal \in dom(rate\_setting2 \rhd \{-1\})$
grd6: $prog1 = null$
grd7: $prog2 = null$

then

act1: $prog := call\_basal\_update$
act2: $par\_basal\_update\_rate\_t := t\_basal$
act3: $prog1 := call\_basal\_update$
act4: $prog2 := call\_basal\_update$

end

Event control5.timer ⟨ordinary⟩ $\widehat{=}$
extends control5.timer

when

grd1:
$\neg($
$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$
$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$
$(time = t\_sd)) \lor$
$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$
$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$
$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$
$)$
grd2: $sdp\_add \neq 1$

then

act1: $time := time + 1$

end

Event NormalBolus.normal_bolus_start_calculate_insulin_needed ⟨ordinary⟩ $\widehat{=}$
extends NormalBolus.normal_bolus_start_calculate_insulin_needed

any

$insulin$

where

NormalBolus.grd1: $insulin > 0$
NormalBolus.grd3: $normal\_add = 0$
grd1: $prog1 = call\_normal\_start$

grd2: $prog2 = call\_normal\_start$

then

NormalBolus.act1: $insulin\_needed := insulin$

NormalBolus.act2: $normal\_add := 1$

end

Event NormalBolus.normal_bolus_start_calculate_lasting_time ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_start_calculate_lasting_time

when

NormalBolus.grd1: $normal\_add = 1$

then

NormalBolus.act1: $normal\_delivering\_time := insulin\_needed/normal\_bolus\_rate$

NormalBolus.act2: $insulin\_needed := 0$

NormalBolus.act3: $normal\_add := 2$

end

Event NormalBolus.normal_bolus_delivery ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_delivery

when

NormalBolus.grd2: $normal\_add = 2$

then

NormalBolus.act1: $normal\_delivering\_rate := normal\_bolus\_rate$

NormalBolus.act2: $normal\_add := 3$

act1: $prog1 := return\_normal\_start$

act2: $prog2 := return\_normal\_start$

end

Event NormalBolus.normal_bolus_suspend ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_suspend

when

grd1: $prog1 = call\_normal\_suspend$

grd2: $prog2 = call\_normal\_suspend$

then

NormalBolus.act1: $normal\_delivering\_rate := 0$

NormalBolus.act2: $normal\_delivering\_time := 0$

NormalBolus.act3: $normal\_add := 0$

NormalBolus.act4: $normal\_bolus\_suspend := TRUE$

act1: $prog1 := return\_normal\_suspend$

act2: $prog2 := return\_normal\_suspend$

end

Event NormalBolus.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_finish

when

grd1: $prog1 = call\_normal\_finish$

grd2: $prog2 = call\_normal\_finish$

then

NormalBolus.act1: $normal\_delivering\_rate := 0$

NormalBolus.act2: $normal\_delivering\_time := 0$

NormalBolus.act3: $normal\_add := 0$

act1: $prog1 := return\_normal\_finish$

act2: $prog2 := return\_normal\_finish$

end

Event NormalBolus.normal_bolus_resume ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_resume

when

grd1: $prog1 = call\_normal\_resume$

grd2: $prog2 = call\_normal\_resume$

then

NormalBolus.act1: $normal\_bolus\_suspend := FALSE$

act1: $normal\_delivering\_rate := 0$

act2: $prog1 := return\_normal\_resume$
act3: $prog2 := return\_normal\_resume$

end

Event Square_Dual_bolus2.start ⟨ordinary⟩ $\widehat{=}$

refines Square_Dual_bolus2.start

any

t
r

where

Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$
Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$
grd2: $prog2 = call\_sd\_start\_s$

then

Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act2: $s\_r := r$
Square_Dual_bolus2.act3: $s\_t := t$
Square_Dual_bolus2.act6: $sd\_module := s$
Square_Dual_bolus2.act7: $d\_deliver\_time := t$
Square_Dual_bolus2.act8: $d\_deliver\_rate := r$
act2: $prog2 := return\_sd\_start\_s$

end

Event Square_Dual_bolus2.start_dual ⟨ordinary⟩ $\widehat{=}$

refines Square_Dual_bolus2.start_dual

any

t
r
td

where

Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$
Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$
Square_Dual_bolus2.grd4: $td \in \mathbb{N}_1$
grd1: $prog2 = call\_sd\_start\_d$

then

Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act2: $s\_r := r$
Square_Dual_bolus2.act3: $s\_t := t$
Square_Dual_bolus2.act6: $d\_deliver\_time := t + td$
Square_Dual_bolus2.act7: $d\_deliver\_rate := normal\_bolus\_rate$
Square_Dual_bolus2.act8: $d\_t := td$
Square_Dual_bolus2.act9: $sd\_module := d$
act1: $prog2 := return\_sd\_start\_d$

end

Event Square_Dual_bolus2.update_to_dual ⟨ordinary⟩ $\widehat{=}$

refines Square_Dual_bolus2.update_to_dual

when

grd1: $prog2 = call\_sd\_update$

then

Square_Dual_bolus2.act2: $d\_deliver\_rate := s\_r$
Square_Dual_bolus2.act3: $sd\_flag := s$
act1: $prog2 := return\_sd\_update$

end

Event Square_Dual_bolus2.finish ⟨ordinary⟩ $\widehat{=}$

refines Square_Dual_bolus2.finish

when

grd1: $prog2 = call\_sd\_finish$

then

Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act4: $s\_r := 0$

Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_finish$

end

Event Square_Dual_bolus2.suspend ⟨ordinary⟩ $\widehat{=}$
refines Square_Dual_bolus2.suspend

when
grd1: $prog2 = call\_sd\_suspend$

then
Square_Dual_bolus2.act1: $state := suspend$
Square_Dual_bolus2.act4: $s\_r := 0$
Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_suspend$

end

Event Square_Dual_bolus2.resume ⟨ordinary⟩ $\widehat{=}$
refines Square_Dual_bolus2.resume

when
grd1: $prog2 = call\_sd\_resume$

then
Square_Dual_bolus2.act1: $state := off$
act1: $prog2 := return\_sd\_resume$

end

Event Square_Dual_bolus2.preempted ⟨ordinary⟩ $\widehat{=}$
refines Square_Dual_bolus2.preempted

when
grd1: $prog2 = call\_sd\_preempt$

with
t: $t = par\_sd\_preempt\_t$

then
Square_Dual_bolus2.act1: $state := preempt$
Square_Dual_bolus2.act4: $d\_deliver\_time := par\_sd\_preempt\_t$
Square_Dual_bolus2.act5: $d\_deliver\_rate := 0$
act1: $prog2 := return\_sd\_preempt$

end

Event Square_Dual_bolus2.resume_from_preempt ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.resume_from_preempt

any
r

where
Square_Dual_bolus2.grd1: $state = preempt$
Square_Dual_bolus2.grd2: $sd\_module = s \Rightarrow r = s\_r$
Square_Dual_bolus2.grd3: $sd\_module = d \wedge sd\_flag = d \Rightarrow r = normal\_bolus\_rate$
Square_Dual_bolus2.grd4: $sd\_module = d \wedge sd\_flag = s \Rightarrow r = s\_r$
grd1: $prog2 = call\_sd\_resume\_preempt$

then
Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act4: $d\_deliver\_rate := r$
act1: $prog2 := return\_sd\_resume\_preempt$

end

Event Basal6·basal_suspend ⟨ordinary⟩ $\widehat{=}$
extends Basal6·basal_suspend

when

grd1: $prog = call\_basal\_suspend$

Basal6.grd3: $prog\_basal = null$

Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in \neq 0$

Basal6.grd2: ⟨theorem⟩ $basal\_mode = delivering$

grd2: $prog1 = call\_basal\_suspend$

grd3: $prog2 = call\_basal\_suspend$

then

Basal6.act1: $basal\_rate\_in := 0$

Basal6.act2: $basal\_mode := suspended$

act1: $prog := return\_basal\_suspend$

act2: $prog1 := return\_basal\_suspend$

act3: $prog2 := return\_basal\_suspend$

end

Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6·change_setting

any

t

r

where

Basal6.grd5: $prog\_basal = null$

Basal6.grd6: $t \in 0 \mathinner{..} c - 1$

Basal6.grd7: $rate\_setting2(t) \neq -1$

Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$

end

Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.delete_setting

any

t

where

Basal6.grd5: $prog\_basal = null$

Basal6.grd2: $basal\_mode \neq suspended$

Basal6.grd6: $t \in 1 \mathinner{..} c - 1$

Basal6.grd7: $rate\_setting2(t) \neq -1$

grd1: $t \neq par\_basal\_update\_rate\_t$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto -1\}$

end

Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.add_setting

any

t

r

where

Basal6.grd9: $prog\_basal = null$

Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$

Basal6.grd4: $basal\_mode \neq suspended$

Basal6.grd5: $t \in 0 \mathinner{..} c - 1$

Basal6.grd6: $rate\_setting2(t) = -1$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$

end

Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_return

when

Basal6.grd8: $prog\_basal = return\_get\_max$

Basal6.grd9: $add\_resume = 2$
grd1: $prog2 = call\_basal\_resume$

then

Basal6.act1: $basal\_rate\_in := max\_value$
Basal6.act2: $basal\_mode := delivering$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $prog\_basal := null$
Basal6.act5: $add\_resume := 0$
act1: $prog := return\_basal\_resume$
act2: $prog1 := return\_basal\_resume$
act3: $prog2 := return\_basal\_resume$

end

Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call

when

grd1: $prog = call\_basal\_resume$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in = 0$
Basal6.grd3: ⟨theorem⟩ $basal\_mode = suspended$
Basal6.grd6: $add\_resume = 0$
Basal6.grd5: $prog\_basal = null$
grd2: $prog1 = call\_basal\_resume$
grd3: $prog2 = call\_basal\_resume$

then

Basal6.act1: $par\_get\_t := par\_basal\_resume\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_resume := 1$

end

Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_resume = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_resume := 2$

end

Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6·rate_update_return

when

Basal6.grd12: $add\_update = 1$
Basal6.grd4: $prog\_basal = return\_get\_min$
grd1: $prog2 = call\_basal\_update$

then

Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$
Basal6.act2: $btime := min\_value - par\_get\_t$
Basal6.act3: $add\_update := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_update$
act6: $prog1 := return\_basal\_update$
act7: $prog2 := return\_basal\_update$

end

Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.rate_update_call

when

grd1: $prog = call\_basal\_update$
Basal6.grd5: ⟨theorem⟩ $basal\_mode = delivering$
Basal6.grd7: ⟨theorem⟩ $rate\_setting2(par\_basal\_update\_rate\_t) \neq -1$
Basal6.grd3: $add\_update = 0$

Basal6.grd2: $prog\_basal = null$
grd2: $prog1 = call\_basal\_update$
grd3: $prog2 = call\_basal\_update$

then

Basal6.act1: $par\_get\_t := par\_basal\_update\_rate\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_update := 1$

end

Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_return

when

Basal6.grd8: $add\_start = 2$
Basal6.grd9: $prog\_basal = return\_get\_max$
grd1: $prog2 = call\_basal\_start$

then

Basal6.act1: $basal\_mode := delivering$
Basal6.act2: $basal\_rate\_in := max\_value$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $add\_start := 0$
act4: $prog\_basal := null$
act5: $prog := return\_basal\_start$
act6: $prog1 := return\_basal\_start$
act7: $prog2 := return\_basal\_start$

end

Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_call

when

grd1: $prog = call\_basal\_start$
Basal6.grd3: $add\_start = 0$
Basal6.grd4: ⟨theorem⟩ $basal\_mode = stop$
Basal6.grd2: $prog\_basal = null$
grd2: $prog1 = call\_basal\_start$
grd3: $prog2 = call\_basal\_start$

then

Basal6.act1: $par\_get\_t := par\_basal\_start\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_start := 1$

end

Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_start = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_start := 2$

end

Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$

extends Basal6.stop

when

Basal6.grd2: $prog\_basal = null$
Basal6.grd1: $basal\_mode = delivering$
grd1: $prog = call\_basal\_stop$
grd2: $prog1 = call\_basal\_stop$
grd3: $prog2 = call\_basal\_stop$

then

Basal6.act1: $basal\_mode := stop$
Basal6.act2: $basal\_rate\_in := 0$

act1: $prog := return\_basal\_stop$
act2: $prog1 := return\_basal\_stop$
act3: $prog2 := return\_basal\_stop$

end

Event Basal6·get_min_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_1

when
Basal6.grd4: $get\_min\_value\_add = 2$
Basal6.grd5: $par\_t = c$

then
Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$

end

Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_2

when
Basal6.grd5: $get\_min\_value\_add = 3$

then
Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$

end

Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_start

when
Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$

then
Basal6.act1: $par\_t := par\_get\_t + 1$
Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$

end

Event Basal6.find_min_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value

when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$

then
Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$

end

Event Basal6.find_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value_2

when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$

then
Basal6.act1: $temp\_min := par\_t$
Basal6.act2: $get\_min\_value\_add := 3$

end

Event Basal6.get_max_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value

when
Basal6.grd2: $get\_max\_value\_add = 2$

then

Basal6.act3: $prog\_basal := return\_get\_max$

Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$

Basal6.act2: $get\_max\_value\_add := 0$

end

Event Basal6.get_max_value_start ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_max_value_start

when

Basal6.grd2: $get\_max\_value\_add = 0$

Basal6.grd3: $prog\_basal = call\_get\_max$

then

Basal6.act1: $get\_max\_start\_t := par\_get\_t$

Basal6.act2: $get\_max\_value\_add := 1$

Basal6.act3: $par\_t\_max := par\_get\_t$

end

Event Basal6·get_max_value_1 ⟨ordinary⟩ $\widehat{=}$

extends Basal6·get_max_value_1

when

Basal6.grd1: $get\_max\_value\_add = 1$

Basal6.grd3: $par\_t\_max \geq 0$

Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$

then

Basal6.act1: $par\_t\_max := par\_t\_max - 1$

end

Event Basal6.get_max_value_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.get_max_value_2

when

Basal6.grd1: $get\_max\_value\_add = 1$

Basal6.grd2: $par\_t\_max \geq 0$

Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$

then

Basal6.act1: $get\_max\_value\_add := 2$

end

END

MACHINE control_Basal6_NormalBolus_2_Square_Dual_bolus2_5
REFINES control_Basal6_NormalBolus_2_Square_Dual_bolus2_4
SEES c_normalbolus_anim,c_prog2_anim,c_basal_anim,c_sd_bolus
VARIABLES

rate_setting2
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
basal_rate_in
basal_mode
btime
par_basal_start_t
par_basal_resume_t
par_basal_update_rate_t
insulin_needed
normal_add
normal_delivering_time
normal_delivering_rate
normal_bolus_suspend
dmodule
d_update_time
state
s_r
s_t
d_deliver_time
d_deliver_rate
d_t
sd_module
sd_flag
prog2
par_sd_preempt_t
sdp_add
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add

par_t_max
prog_basal
par_get_t
add_resume
add_update
add_start

EVENTS

Initialisation

begin

act1: $normal\_bolus\_work := FALSE$
act2: $sd\_bolus\_work := FALSE$
act3: $sd\_preempted\_by\_normal := FALSE$
act7: $sd\_suspend := FALSE$
act8: $normal\_suspend := FALSE$
act9: $basal\_work := FALSE$
act10: $basal\_suspend := FALSE$
act11: $pump\_rate := 0$
act12: $basal\_rate := 0$
act13: $normal\_rate := 0$
act14: $sd\_rate := 0$
act15: $time := 0$
act16: $t\_basal := 0$
act17: $t\_normal := 0$
act18: $t\_sd := 0$
act19: $dmodule := FALSE$
act20: $d\_update\_time := 0$
Basal1.act4: $btime := c$
Basal1.act2: $basal\_rate\_in := 0$
Basal1.act3: $basal\_mode := stop$
Basal6.act15: $prog\_basal := null$
Basal6.act16: $par\_get\_t := 0$
Basal6.act17: $add\_resume := 0$
Basal6.act18: $add\_update := 0$
Basal6.act19: $add\_start := 0$
Basal6.act5: $rate\_setting2 := (1 \mathinner{..} c - 1 \times \{-1\}) \cup \{0 \mapsto 0\}$
Basal6.act6: $min\_value := 0$
Basal6.act7: $max\_value := 0$
Basal6.act11: $get\_min\_value\_add := 0$
Basal6.act8: $par\_t := 0$
Basal6.act9: $temp\_min := 0$
Basal6.act10: $get\_min\_start\_t := 0$
Basal6.act12: $get\_max\_start\_t := 0$
Basal6.act13: $get\_max\_value\_add := 0$
Basal6.act14: $par\_t\_max := 0$
act22: $par\_basal\_start\_t := 0$
act23: $par\_basal\_resume\_t := 0$
act24: $par\_basal\_update\_rate\_t := 0$
NormalBolus.act1: $insulin\_needed := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_delivering\_rate := 0$
NormalBolus.act4: $normal\_add := 0$
NormalBolus.act5: $normal\_bolus\_suspend := FALSE$
Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act2: $s\_r := 0$
Square_Dual_bolus2.act3: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_module := s$

Square_Dual_bolus2.act10: $sd\_flag := d$
act32: $prog2 := null$
act33: $par\_sd\_preempt\_t := 0$
act34: $sdp\_add := 0$

end

Event control5·normal_bolus_start_1_return ⟨ordinary⟩ $\widehat{=}$

refines control5·normal_bolus_start_1_return

any

t2

where

grd6: $normal\_bolus\_work = FALSE$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \wedge time \neq d\_update\_time$
control5.grd2: $sd\_bolus\_work = TRUE$
control5.grd3: $sd\_preempted\_by\_normal = FALSE$
control5.grd4: $sd\_suspend = FALSE$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$
grd9: $prog2 = return\_normal\_start$
grd4: ⟨theorem⟩ $sdp\_add = 1$

then

control5.act6: $t\_normal := time + normal\_delivering\_time$
control5.act7: $t\_sd := t\_sd - time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $sd\_preempted\_by\_normal := TRUE$
control5.act3: $normal\_rate := normal\_delivering\_rate$
control5.act4: $sd\_rate := 0$
control5.act5: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act8: $d\_update\_time := t2$
act2: $prog2 := null$
act3: $sdp\_add := 0$

end

Event control5·normal_bolus_start_1_call_2 ⟨ordinary⟩ $\widehat{=}$

refines control5·normal_bolus_start_1_call_2

when

grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_preempt$
grd8: ⟨theorem⟩ $sdp\_add = 1$

then

act2: $prog2 := call\_normal\_start$

end

Event control5·normal_bolus_start_1_call_sd_preempt ⟨ordinary⟩ $\widehat{=}$

extends control5 ·normal_bolus_start_1_call_sd_preempt

when

grd1: $prog2 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $dmodule = TRUE \Rightarrow time \neq d\_update\_time$
grd7: $sdp\_add = 0$

then

act1: $prog2 := call\_sd\_preempt$
act2: $par\_sd\_preempt\_t := t\_sd - time$
act3: $sdp\_add := 1$

end

Event control5.normal_bolus_start_2_return ⟨ordinary⟩ $\widehat{=}$

refines control5.normal_bolus_start_2_return

when

control5.grd2: $sd\_bolus\_work = FALSE$

grd2: $prog2 = return\_normal\_start$

then

control5·act4: $t\_normal := time + normal\_delivering\_time$

control5.act1: $normal\_bolus\_work := TRUE$

control5.act2: $normal\_rate := normal\_delivering\_rate$

control5.act3: $pump\_rate := normal\_delivering\_rate + basal\_rate$

act2: $prog2 := null$

end

Event control5·normal_bolus_start_2_call ⟨ordinary⟩ $\widehat{=}$

refines control5·normal_bolus_start_2_call

when

grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_bolus\_work = FALSE$

grd4: $normal\_suspend = FALSE$

grd5: $prog2 = null$

then

act2: $prog2 := call\_normal\_start$

end

Event control5·normal_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$

refines control5·normal_bolus_finish_return

when

grd2: $prog2 = return\_normal\_finish$

then

control5.act4: $t\_normal := 0$

control5.act1: $normal\_bolus\_work := FALSE$

control5.act2: $normal\_rate := normal\_delivering\_rate$

control5.act3: $pump\_rate := basal\_rate$

act2: $prog2 := null$

end

Event control5·normal_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$

refines control5·normal_bolus_finish_call

when

grd2: $time = t\_normal$

grd3: $normal\_bolus\_work = TRUE$

grd4: $normal\_suspend = FALSE$

grd5: $prog2 = null$

then

act2: $prog2 := call\_normal\_finish$

end

Event control5·normal_suspend_return ⟨ordinary⟩ $\widehat{=}$

refines control5·normal_suspend_return

when

grd2: $prog2 = return\_normal\_suspend$

then

control5.act4: $t\_normal := t\_normal - time$

control5.act1: $normal\_suspend := TRUE$

control5.act2: $normal\_rate := normal\_delivering\_rate$

control5.act3: $pump\_rate := basal\_rate$

act3: $prog2 := null$

end

Event control5·normal_bolus_suspend_call ⟨ordinary⟩ $\widehat{=}$

refines control5·normal_bolus_suspend_call

when

grd2: $normal\_bolus\_work = TRUE$

grd3: $normal\_suspend = FALSE$
grd4: $prog2 = null$
then
act2: $prog2 := call\_normal\_suspend$
end

Event control5·normal_resume_return ⟨ordinary⟩ $\widehat{=}$
refines control5·normal_resume_return
when
grd2: $prog2 = return\_normal\_resume$
then
control5.act4: $t\_normal := 0$
control5.act1: $normal\_suspend := FALSE$
control5.act2: $normal\_rate := 0$
control5.act3: $pump\_rate := basal\_rate$
act2: $normal\_bolus\_work := FALSE$
act3: $prog2 := null$
end

Event control5·normal_bolus_resume_call ⟨ordinary⟩ $\widehat{=}$
refines control5·normal_bolus_resume_call
when
grd2: $normal\_suspend = TRUE$
grd3: $prog2 = null$
then
act2: $prog2 := call\_normal\_resume$
end

Event control5·square_or_dual_bolus_start_s_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·square_or_dual_bolus_start_s_return
when
grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_start\_s$
then
act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + d\_deliver\_time$
act5: $dmodule := FALSE$
act6: $prog2 := null$
end

Event control5square_or_dual_bolus_start_s_call ⟨ordinary⟩ $\widehat{=}$
extends control5square_or_dual_bolus_start_s_call
when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $prog2 := call\_sd\_start\_s$
end

Event control5·square_or_dual_bolus_start_d_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·square_or_dual_bolus_start_d_return
when
grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd8: $prog2 = return\_sd\_start\_d$

then
act1: $sd\_bolus\_work := TRUE$
act5: $dmodule := TRUE$
act6: $d\_update\_time := time + d\_t$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + d\_deliver\_time$
act7: $prog2 := null$
end

Event control5·square_or_dual_bolus_start_d_call ⟨ordinary⟩ $\widehat{=}$
extends control5·square_or_dual_bolus_start_d_call
when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $prog2 := call\_sd\_start\_d$
end

Event control5.square_or_dual_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_finish_return
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$
grd6: $prog2 = return\_sd\_finish$
then
act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := d\_deliver\_time$
act5: $dmodule := FALSE$
act6: $prog2 := null$
end

Event control5.square_or_dual_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_finish_call
when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$
then
act1: $prog2 := call\_sd\_finish$
end

Event control5.square_or_dual_bolus_resume_from_normal_return ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_resume_from_normal_return
any
$t2$
where
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_resume\_preempt$
grd7: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

then
act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $prog2 := null$
act6: $d\_update\_time := t2$
end

Event control5.square_or_dual_bolus_resume_from_normal_call ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_resume_from_normal_call
when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = TRUE$
grd4: $normal\_bolus\_work = FALSE$
grd5: $sd\_suspend = FALSE$
then
act1: $prog2 := call\_sd\_resume\_preempt$
end

Event control5.sd_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_suspend_return
when
grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $prog2 = return\_sd\_suspend$
then
act1: $sd\_suspend := TRUE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$
act6: $prog2 := null$
act7: $d\_update\_time := 0$
end

Event control5.sd_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_suspend_call
when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $prog2 := call\_sd\_suspend$
end

Event control5.square_or_dual_update_rate_return ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_update_rate_return
when
grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd6: $dmodule = TRUE$
grd7: $time = d\_update\_time$
grd8: $prog2 = return\_sd\_update$
then
act1: $sd\_rate := d\_deliver\_rate$
act2: $pump\_rate := d\_deliver\_rate + basal\_rate$
act3: $dmodule := FALSE$

act4: $prog2 := null$
act5: $d\_update\_time := 0$
end
Event control5·square_or_dual_update_rate_call ⟨ordinary⟩ $\widehat{=}$
extends control5·square_or_dual_update_rate_call
when
grd1: $prog2 = null$
grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $dmodule = TRUE$
grd6: $time = d\_update\_time$
then
act1: $prog2 := call\_sd\_update$
end
Event control5.sd_resume_return ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_resume_return
when
grd1: $sd\_suspend = TRUE$
grd3: $prog2 = return\_sd\_resume$
then
act1: $sd\_suspend := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $prog2 := null$
act6: $sd\_bolus\_work := FALSE$
end
Event control5.sd_resume_call ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_resume_call
when
grd1: $prog2 = null$
grd2: $sd\_suspend = TRUE$
then
act1: $prog2 := call\_sd\_resume$
end
Event control5·basal_start_return ⟨ordinary⟩ $\widehat{=}$
refines control5·basal_start_return
when
grd3: $prog2 = return\_basal\_start$
then
control5.act4: $t\_basal := time + btime$
control5.act1: $basal\_work := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_start_call ⟨ordinary⟩ $\widehat{=}$
refines control5·basal_start_call
when
grd1: $basal\_work = FALSE$
grd2: $basal\_suspend = FALSE$
grd5: $prog2 = null$
then
act2: $par\_basal\_start\_t := timemodc$
act4: $prog2 := call\_basal\_start$
end

Event control5·basal_stop_return ⟨ordinary⟩ $\widehat{=}$
refines control5·basal_stop_return
when
grd3: $prog2 = return\_basal\_stop$
then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_work := FALSE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_stop_call ⟨ordinary⟩ $\widehat{=}$
refines control5·basal_stop_call
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd5: $prog2 = null$
then
act3: $prog2 := call\_basal\_stop$
end
Event control5·basal_suspend_return ⟨ordinary⟩ $\widehat{=}$
refines control5·basal_suspend_return
when
grd3: $prog2 = return\_basal\_suspend$
then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_suspend := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_suspend_call ⟨ordinary⟩ $\widehat{=}$
refines control5·basal_suspend_call
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

grd4: $basal\_rate \neq 0$
grd6: $prog2 = null$
then
act3: $prog2 := call\_basal\_suspend$
end
Event control5·basal_resume_return ⟨ordinary⟩ $\widehat{=}$
refines control5·basal_resume_return
when
grd3: $prog2 = return\_basal\_resume$
then
control5.act1: $basal\_suspend := FALSE$
control5.act4: $t\_basal := btime + time$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_resume_call ⟨ordinary⟩ $\widehat{=}$
refines control5·basal_resume_call
when
grd1: $basal\_suspend = TRUE$

grd4: $prog2 = null$

then

act2: $par\_basal\_resume\_t := timemodc$

act4: $prog2 := call\_basal\_resume$

end

Event control5·basal_update_rate_return ⟨ordinary⟩ $\widehat{=}$

refines control5·basal_update_rate_return

when

grd3: $prog2 = return\_basal\_update$

then

control5.act3: $t\_basal := time + btime$

control5.act1: $basal\_rate := basal\_rate\_in$

control5.act2: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$

act3: $prog2 := null$

end

Event control5·basal_update_rate_call ⟨ordinary⟩ $\widehat{=}$

refines control5 ·basal_update_rate_call

when

grd1: $t\_basal = time$

grd2: $basal\_suspend = FALSE$

grd3: $basal\_work = TRUE$

grd5: $t\_basal \in dom(rate\_setting2 \rhd \{-1\})$

grd7: $prog2 = null$

then

act2: $par\_basal\_update\_rate\_t := t\_basal$

act4: $prog2 := call\_basal\_update$

end

Event control5.timer ⟨ordinary⟩ $\widehat{=}$

extends control5.timer

when

grd1:

$\neg($

$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$

$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$

$(time = t\_sd)) \lor$

$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$

$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$

$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$

$)$

grd2: $sdp\_add \neq 1$

then

act1: $time := time + 1$

end

Event NormalBolus.normal_bolus_start_calculate_insulin_needed ⟨ordinary⟩ $\widehat{=}$

refines NormalBolus.normal_bolus_start_calculate_insulin_needed

any

insulin

where

NormalBolus.grd1: $insulin > 0$

NormalBolus.grd3: $normal\_add = 0$

grd2: $prog2 = call\_normal\_start$

then

NormalBolus.act1: $insulin\_needed := insulin$

NormalBolus.act2: $normal\_add := 1$

end

Event NormalBolus.normal_bolus_start_calculate_lasting_time ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_start_calculate_lasting_time

when
NormalBolus.grd1: $normal\_add = 1$
then
NormalBolus.act1: $normal\_delivering\_time := insulin\_needed / normal\_bolus\_rate$
NormalBolus.act2: $insulin\_needed := 0$
NormalBolus.act3: $normal\_add := 2$
end

Event NormalBolus.normal_bolus_delivery ⟨ordinary⟩ $\widehat{=}$
refines NormalBolus.normal_bolus_delivery
when
NormalBolus.grd2: $normal\_add = 2$
then
NormalBolus.act1: $normal\_delivering\_rate := normal\_bolus\_rate$
NormalBolus.act2: $normal\_add := 3$
act2: $prog2 := return\_normal\_start$
end

Event NormalBolus.normal_bolus_suspend ⟨ordinary⟩ $\widehat{=}$
refines NormalBolus.normal_bolus_suspend
when
grd2: $prog2 = call\_normal\_suspend$
then
NormalBolus.act1: $normal\_delivering\_rate := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_add := 0$
NormalBolus.act4: $normal\_bolus\_suspend := TRUE$
act2: $prog2 := return\_normal\_suspend$
end

Event NormalBolus.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$
refines NormalBolus.normal_bolus_finish
when
grd2: $prog2 = call\_normal\_finish$
then
NormalBolus.act1: $normal\_delivering\_rate := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_add := 0$
act2: $prog2 := return\_normal\_finish$
end

Event NormalBolus.normal_bolus_resume ⟨ordinary⟩ $\widehat{=}$
refines NormalBolus.normal_bolus_resume
when
grd2: $prog2 = call\_normal\_resume$
then
NormalBolus.act1: $normal\_bolus\_suspend := FALSE$
act2: $prog2 := return\_normal\_resume$
act3: $normal\_delivering\_rate := 0$
end

Event Square_Dual_bolus2.start ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.start
any
t
r
where
Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$
Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$
grd2: $prog2 = call\_sd\_start\_s$
then
Square_Dual_bolus2.act1: $state := deliver$

Square_Dual_bolus2.act2: $s\_r := r$
Square_Dual_bolus2.act3: $s\_t := t$
Square_Dual_bolus2.act6: $sd\_module := s$
Square_Dual_bolus2.act7: $d\_deliver\_time := t$
Square_Dual_bolus2.act8: $d\_deliver\_rate := r$
act2: $prog2 := return\_sd\_start\_s$

end

Event Square_Dual_bolus2.start_dual ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.start_dual

any

t
r
td

where

Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$
Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$
Square_Dual_bolus2.grd4: $td \in \mathbb{N}_1$
grd1: $prog2 = call\_sd\_start\_d$

then

Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act2: $s\_r := r$
Square_Dual_bolus2.act3: $s\_t := t$
Square_Dual_bolus2.act6: $d\_deliver\_time := t + td$
Square_Dual_bolus2.act7: $d\_deliver\_rate := normal\_bolus\_rate$
Square_Dual_bolus2.act8: $d\_t := td$
Square_Dual_bolus2.act9: $sd\_module := d$
act1: $prog2 := return\_sd\_start\_d$

end

Event Square_Dual_bolus2.update_to_dual ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.update_to_dual

when

grd1: $prog2 = call\_sd\_update$

then

Square_Dual_bolus2.act2: $d\_deliver\_rate := s\_r$
Square_Dual_bolus2.act3: $sd\_flag := s$
act1: $prog2 := return\_sd\_update$

end

Event Square_Dual_bolus2.finish ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.finish

when

grd1: $prog2 = call\_sd\_finish$

then

Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act4: $s\_r := 0$
Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_finish$

end

Event Square_Dual_bolus2.suspend ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.suspend

when

grd1: $prog2 = call\_sd\_suspend$

then

Square_Dual_bolus2.act1: $state := suspend$
Square_Dual_bolus2.act4: $s\_r := 0$

Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_suspend$

end

Event Square_Dual_bolus2.resume ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.resume

when

grd1: $prog2 = call\_sd\_resume$

then

Square_Dual_bolus2.act1: $state := off$
act1: $prog2 := return\_sd\_resume$

end

Event Square_Dual_bolus2.preempted ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.preempted

when

grd1: $prog2 = call\_sd\_preempt$

then

Square_Dual_bolus2.act1: $state := preempt$
Square_Dual_bolus2.act4: $d\_deliver\_time := par\_sd\_preempt\_t$
Square_Dual_bolus2.act5: $d\_deliver\_rate := 0$
act1: $prog2 := return\_sd\_preempt$

end

Event Square_Dual_bolus2.resume_from_preempt ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.resume_from_preempt

any

r

where

Square_Dual_bolus2.grd1: $state = preempt$
Square_Dual_bolus2.grd2: $sd\_module = s \Rightarrow r = s\_r$
Square_Dual_bolus2.grd3: $sd\_module = d \wedge sd\_flag = d \Rightarrow r = normal\_bolus\_rate$
Square_Dual_bolus2.grd4: $sd\_module = d \wedge sd\_flag = s \Rightarrow r = s\_r$
grd1: $prog2 = call\_sd\_resume\_preempt$

then

Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act4: $d\_deliver\_rate := r$
act1: $prog2 := return\_sd\_resume\_preempt$

end

Event Basal6·basal_suspend ⟨ordinary⟩ $\widehat{=}$
refines Basal6·basal_suspend

when

Basal6.grd3: $prog\_basal = null$
grd3: $prog2 = call\_basal\_suspend$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in \neq 0$
Basal6.grd2: ⟨theorem⟩ $basal\_mode = delivering$

then

Basal6.act1: $basal\_rate\_in := 0$
Basal6.act2: $basal\_mode := suspended$
act3: $prog2 := return\_basal\_suspend$

end

Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6·change_setting

any

t
r

where

Basal6.grd5: $prog\_basal = null$

Basal6.grd6: $t \in 0 \mathinner{..} c - 1$

Basal6.grd7: $rate\_setting2(t) \neq -1$

Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!- \{t \mapsto r\}$

end

Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.delete_setting

any

t

where

Basal6.grd5: $prog\_basal = null$

Basal6.grd2: $basal\_mode \neq suspended$

Basal6.grd6: $t \in 1 \mathinner{..} c - 1$

Basal6.grd7: $rate\_setting2(t) \neq -1$

grd1: $t \neq par\_basal\_update\_rate\_t$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!- \{t \mapsto -1\}$

end

Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.add_setting

any

t

r

where

Basal6.grd9: $prog\_basal = null$

Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$

Basal6.grd4: $basal\_mode \neq suspended$

Basal6.grd5: $t \in 0 \mathinner{..} c - 1$

Basal6.grd6: $rate\_setting2(t) = -1$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!\!- \{t \mapsto r\}$

end

Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$

refines Basal6.basal_resume_return

when

Basal6.grd8: $prog\_basal = return\_get\_max$

Basal6.grd9: $add\_resume = 2$

grd1: $prog2 = call\_basal\_resume$

then

Basal6.act1: $basal\_rate\_in := max\_value$

Basal6.act2: $basal\_mode := delivering$

Basal6.act3: $btime := min\_value - par\_get\_t$

Basal6.act4: $prog\_basal := null$

Basal6.act5: $add\_resume := 0$

act3: $prog2 := return\_basal\_resume$

end

Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$

refines Basal6.basal_resume_call

when

Basal6.grd6: $add\_resume = 0$

Basal6.grd5: $prog\_basal = null$

grd3: $prog2 = call\_basal\_resume$

Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in = 0$

Basal6.grd3: ⟨theorem⟩ $basal\_mode = suspended$

then

Basal6.act1: $par\_get\_t := par\_basal\_resume\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_resume := 1$

end

Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_resume = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_resume := 2$

end

Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$

refines Basal6·rate_update_return

when

Basal6.grd12: $add\_update = 1$
Basal6.grd4: $prog\_basal = return\_get\_min$
grd1: $prog2 = call\_basal\_update$

then

Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$
Basal6.act2: $btime := min\_value - par\_get\_t$
Basal6.act3: $add\_update := 0$
act4: $prog\_basal := null$
act7: $prog2 := return\_basal\_update$

end

Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$

refines Basal6.rate_update_call

when

Basal6.grd3: $add\_update = 0$
Basal6.grd2: $prog\_basal = null$
grd3: $prog2 = call\_basal\_update$
Basal6.grd5: ⟨theorem⟩ $basal\_mode = delivering$
Basal6.grd7: ⟨theorem⟩ $rate\_setting2(par\_basal\_update\_rate\_t) \neq -1$

then

Basal6.act1: $par\_get\_t := par\_basal\_update\_rate\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_update := 1$

end

Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$

refines Basal6.start_return

when

Basal6.grd8: $add\_start = 2$
Basal6.grd9: $prog\_basal = return\_get\_max$
grd1: $prog2 = call\_basal\_start$

then

Basal6.act1: $basal\_mode := delivering$
Basal6.act2: $basal\_rate\_in := max\_value$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $add\_start := 0$
act4: $prog\_basal := null$
act7: $prog2 := return\_basal\_start$

end

Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$

refines Basal6.start_call

when

Basal6.grd3: $add\_start = 0$

Basal6.grd2: $prog\_basal = null$
grd3: $prog2 = call\_basal\_start$
Basal6.grd4: ⟨theorem⟩ $basal\_mode = stop$
then
Basal6.act1: $par\_get\_t := par\_basal\_start\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_start := 1$
end
Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call_2
when
Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_start = 1$
then
Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_start := 2$
end
Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$
refines Basal6.stop
when
Basal6.grd2: $prog\_basal = null$
Basal6.grd1: $basal\_mode = delivering$
grd3: $prog2 = call\_basal\_stop$
then
Basal6.act1: $basal\_mode := stop$
Basal6.act2: $basal\_rate\_in := 0$
act3: $prog2 := return\_basal\_stop$
end
Event Basal6·get_min_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6·get_min_value_1
when
Basal6.grd4: $get\_min\_value\_add = 2$
Basal6.grd5: $par\_t = c$
then
Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_2
when
Basal6.grd5: $get\_min\_value\_add = 3$
then
Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_start
when
Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$
then
Basal6.act1: $par\_t := par\_get\_t + 1$
Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$
end

Event Basal6.find_min_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$
then
Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$
end
Event Basal6.find_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value_2
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$
then
Basal6.act1: $temp\_min := par\_t$
Basal6.act2: $get\_min\_value\_add := 3$
end
Event Basal6.get_max_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value
when
Basal6.grd2: $get\_max\_value\_add = 2$
then
Basal6.act3: $prog\_basal := return\_get\_max$
Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$
Basal6.act2: $get\_max\_value\_add := 0$
end
Event Basal6.get_max_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_start
when
Basal6.grd2: $get\_max\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_max$
then
Basal6.act1: $get\_max\_start\_t := par\_get\_t$
Basal6.act2: $get\_max\_value\_add := 1$
Basal6.act3: $par\_t\_max := par\_get\_t$
end
Event Basal6·get_max_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6·get_max_value_1
when
Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd3: $par\_t\_max \geq 0$
Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$
then
Basal6.act1: $par\_t\_max := par\_t\_max - 1$
end
Event Basal6.get_max_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_2
when
Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd2: $par\_t\_max \geq 0$
Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$
then
Basal6.act1: $get\_max\_value\_add := 2$
end
END

MACHINE control_Basal6_NormalBolus_2_Square_Dual_bolus2_5_c
REFINES control_Basal6_NormalBolus_2_Square_Dual_bolus2_5
SEES c_normalbolus_anim,c_prog2_anim,c_basal_anim,c_sd_bolus
VARIABLES

rate_setting2
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
basal_rate_in
basal_mode
btime
par_basal_start_t
par_basal_resume_t
par_basal_update_rate_t
insulin_needed
normal_add
normal_delivering_time
normal_delivering_rate
normal_bolus_suspend
dmodule
d_update_time
state
s_r
s_t
d_deliver_time
d_deliver_rate
d_t
sd_module
sd_flag
prog2
par_sd_preempt_t
sdp_add
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add

par_t_max
prog_basal
par_get_t
add_resume
add_update
add_start
fbegin
fend
rate_basal_c
normal_delivering_rate_c
nb_now
nb_new_now
sd_now
sd_new_now
sd_rate_c

INVARIANTS

inv11: $sd\_rate\_c \in \mathbb{N} \pfun \mathbb{N}$
inv12: $sd\_now \in dom(sd\_rate\_c)$
inv13: $sd\_new\_now \in \mathbb{N}$
inv14: $d\_deliver\_rate = sd\_rate\_c(sd\_now)$
inv10: $state \in \{deliver, preempt\} \land sd\_module = d \land sd\_flag = d \Rightarrow d\_deliver\_time > s\_t$
inv7: $normal\_delivering\_rate\_c \in \mathbb{N} \pfun 0 \upto normal\_bolus\_rate$
inv8: $nb\_new\_now \in \mathbb{N}$
inv9: $nb\_now \in dom(normal\_delivering\_rate\_c)$
inv1: $normal\_delivering\_rate\_c(nb\_now) = normal\_delivering\_rate$
inv2: $rate\_basal\_c \in \mathbb{N} \pfun 0 \upto basal\_max$
inv3: $fbegin \in dom(rate\_basal\_c)$
inv4: $fend \in 0 \upto c$
inv5: $rate\_basal\_c(fbegin) = basal\_rate\_in$
inv6: $basal\_mode = delivering \Rightarrow fend > fbegin$

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$
act2: $sd\_bolus\_work := FALSE$
act3: $sd\_preempted\_by\_normal := FALSE$
act7: $sd\_suspend := FALSE$
act8: $normal\_suspend := FALSE$
act9: $basal\_work := FALSE$
act10: $basal\_suspend := FALSE$
act11: $pump\_rate := 0$
act12: $basal\_rate := 0$
act13: $normal\_rate := 0$
act14: $sd\_rate := 0$
act15: $time := 0$
act16: $t\_basal := 0$
act17: $t\_normal := 0$
act18: $t\_sd := 0$
act19: $dmodule := FALSE$
act20: $d\_update\_time := 0$
Basal1.act4: $btime := c$
Basal1.act2: $basal\_rate\_in := 0$
Basal1.act3: $basal\_mode := stop$
Basal6.act15: $prog\_basal := null$
Basal6.act16: $par\_get\_t := 0$

Basal6.act17: $add\_resume := 0$
Basal6.act18: $add\_update := 0$
Basal6.act19: $add\_start := 0$
Basal6.act5: $rate\_setting2 := (1 \mathinner{..} c - 1 \times \{-1\}) \cup \{0 \mapsto 0\}$
Basal6.act6: $min\_value := 0$
Basal6.act7: $max\_value := 0$
Basal6.act11: $get\_min\_value\_add := 0$
Basal6.act8: $par\_t := 0$
Basal6.act9: $temp\_min := 0$
Basal6.act10: $get\_min\_start\_t := 0$
Basal6.act12: $get\_max\_start\_t := 0$
Basal6.act13: $get\_max\_value\_add := 0$
Basal6.act14: $par\_t\_max := 0$
act22: $par\_basal\_start\_t := 0$
act23: $par\_basal\_resume\_t := 0$
act24: $par\_basal\_update\_rate\_t := 0$
NormalBolus.act1: $insulin\_needed := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_delivering\_rate := 0$
NormalBolus.act4: $normal\_add := 0$
NormalBolus.act5: $normal\_bolus\_suspend := FALSE$
Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act2: $s\_r := 0$
Square_Dual_bolus2.act3: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_module := s$
Square_Dual_bolus2.act10: $sd\_flag := d$
act32: $prog2 := null$
act33: $par\_sd\_preempt\_t := 0$
act34: $sdp\_add := 0$
act35: $fbegin := 0$
act36: $fend := 0$
act37: $rate\_basal\_c := \{0 \mapsto 0\}$
act38: $normal\_delivering\_rate\_c := \{0 \mapsto 0\}$
act39: $nb\_now := 0$
act40: $nb\_new\_now := 0$
act41: $sd\_now := 0$
act42: $sd\_new\_now := 0$
act43: $sd\_rate\_c := \{0 \mapsto 0\}$

end

Event control5·normal_bolus_start_1_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_1_return

any

$t2$

where

grd6: $normal\_bolus\_work = FALSE$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \wedge time \neq d\_update\_time$
control5.grd2: $sd\_bolus\_work = TRUE$
control5.grd3: $sd\_preempted\_by\_normal = FALSE$
control5.grd4: $sd\_suspend = FALSE$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$
grd9: $prog2 = return\_normal\_start$
grd4: ⟨theorem⟩ $sdp\_add = 1$

then

control5.act6: $t\_normal := time + normal\_delivering\_time$
control5.act7: $t\_sd := t\_sd - time$
control5.act1: $normal\_bolus\_work := TRUE$

control5.act2: $sd\_preempted\_by\_normal := TRUE$
control5.act3: $normal\_rate := normal\_delivering\_rate$
control5.act4: $sd\_rate := 0$
control5.act5: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act8: $d\_update\_time := t2$
act2: $prog2 := null$
act3: $sdp\_add := 0$

end

Event control5·normal_bolus_start_1_call_2 ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_1_call_2

when

grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_preempt$
grd8: ⟨theorem⟩ $sdp\_add = 1$

then

act2: $prog2 := call\_normal\_start$

end

Event control5·normal_bolus_start_1_call_sd_preempt ⟨ordinary⟩ $\widehat{=}$
extends control5 ·normal_bolus_start_1_call_sd_preempt

when

grd1: $prog2 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $dmodule = TRUE \Rightarrow time \neq d\_update\_time$
grd7: $sdp\_add = 0$

then

act1: $prog2 := call\_sd\_preempt$
act2: $par\_sd\_preempt\_t := t\_sd - time$
act3: $sdp\_add := 1$

end

Event control5.normal_bolus_start_2_return ⟨ordinary⟩ $\widehat{=}$
extends control5.normal_bolus_start_2_return

when

control5.grd2: $sd\_bolus\_work = FALSE$
grd2: $prog2 = return\_normal\_start$

then

control5·act4: $t\_normal := time + normal\_delivering\_time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act2: $prog2 := null$

end

Event control5·normal_bolus_start_2_call ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_2_call

when

grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = FALSE$
grd4: $normal\_suspend = FALSE$
grd5: $prog2 = null$

then

act2: $prog2 := call\_normal\_start$

end

Event control5·normal_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_bolus_finish_return
when
grd2: $prog2 = return\_normal\_finish$
then
control5.act4: $t\_normal := 0$
control5.act1: $normal\_bolus\_work := FALSE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act2: $prog2 := null$
end
Event control5·normal_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$
extends control5 ·normal_bolus_finish_call
when
grd2: $time = t\_normal$
grd3: $normal\_bolus\_work = TRUE$
grd4: $normal\_suspend = FALSE$
grd5: $prog2 = null$
then
act2: $prog2 := call\_normal\_finish$
end
Event control5·normal_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_suspend_return
when
grd2: $prog2 = return\_normal\_suspend$
then
control5.act4: $t\_normal := t\_normal - time$
control5.act1: $normal\_suspend := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act3: $prog2 := null$
end
Event control5·normal_bolus_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_suspend_call
when
grd2: $normal\_bolus\_work = TRUE$
grd3: $normal\_suspend = FALSE$
grd4: $prog2 = null$
then
act2: $prog2 := call\_normal\_suspend$
end
Event control5·normal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_resume_return
when
grd2: $prog2 = return\_normal\_resume$
then
control5.act4: $t\_normal := 0$
control5.act1: $normal\_suspend := FALSE$
control5.act2: $normal\_rate := 0$
control5.act3: $pump\_rate := basal\_rate$
act2: $normal\_bolus\_work := FALSE$
act3: $prog2 := null$
end
Event control5·normal_bolus_resume_call ⟨ordinary⟩ $\widehat{=}$
extends control5 ·normal_bolus_resume_call
when
grd2: $normal\_suspend = TRUE$
grd3: $prog2 = null$

then
act2: $prog2 := call\_normal\_resume$
end
Event control5·square_or_dual_bolus_start_s_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·square_or_dual_bolus_start_s_return
when
grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_start\_s$
then
act1: $sd\_bolus\_work := TRUE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + d\_deliver\_time$
act5: $dmodule := FALSE$
act6: $prog2 := null$
end
Event control5square_or_dual_bolus_start_s_call ⟨ordinary⟩ $\widehat{=}$
extends control5square_or_dual_bolus_start_s_call
when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $prog2 := call\_sd\_start\_s$
end
Event control5·square_or_dual_bolus_start_d_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·square_or_dual_bolus_start_d_return
when
grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd8: $prog2 = return\_sd\_start\_d$
then
act1: $sd\_bolus\_work := TRUE$
act5: $dmodule := TRUE$
act6: $d\_update\_time := time + d\_t$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + d\_deliver\_time$
act7: $prog2 := null$
end
Event control5·square_or_dual_bolus_start_d_call ⟨ordinary⟩ $\widehat{=}$
extends control5·square_or_dual_bolus_start_d_call
when
grd1: $prog2 = null$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
then
act1: $prog2 := call\_sd\_start\_d$
end
Event control5.square_or_dual_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$
extends control5.square_or_dual_bolus_finish_return
when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$
grd6: $prog2 = return\_sd\_finish$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := d\_deliver\_time$
act5: $dmodule := FALSE$
act6: $prog2 := null$

end

Event control5.square_or_dual_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_bolus_finish_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$

then

act1: $prog2 := call\_sd\_finish$

end

Event control5.square_or_dual_bolus_resume_from_normal_return ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_bolus_resume_from_normal_return

any

$t2$

where

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_resume\_preempt$
grd7: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $prog2 := null$
act6: $d\_update\_time := t2$

end

Event control5.square_or_dual_bolus_resume_from_normal_call ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_bolus_resume_from_normal_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = TRUE$
grd4: $normal\_bolus\_work = FALSE$
grd5: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_resume\_preempt$

end

Event control5.sd_suspend_return ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_suspend_return

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $prog2 = return\_sd\_suspend$

then

act1: $sd\_suspend := TRUE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$
act6: $prog2 := null$
act7: $d\_update\_time := 0$

end

Event control5.sd_suspend_call ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_suspend_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_suspend$

end

Event control5.square_or_dual_update_rate_return ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_update_rate_return

when

grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd6: $dmodule = TRUE$
grd7: $time = d\_update\_time$
grd8: $prog2 = return\_sd\_update$

then

act1: $sd\_rate := d\_deliver\_rate$
act2: $pump\_rate := d\_deliver\_rate + basal\_rate$
act3: $dmodule := FALSE$
act4: $prog2 := null$
act5: $d\_update\_time := 0$

end

Event control5·square_or_dual_update_rate_call ⟨ordinary⟩ $\widehat{=}$

extends control5·square_or_dual_update_rate_call

when

grd1: $prog2 = null$
grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $dmodule = TRUE$
grd6: $time = d\_update\_time$

then

act1: $prog2 := call\_sd\_update$

end

Event control5.sd_resume_return ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_resume_return

when

grd1: $sd\_suspend = TRUE$
grd3: $prog2 = return\_sd\_resume$

then

act1: $sd\_suspend := FALSE$

act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $prog2 := null$
act6: $sd\_bolus\_work := FALSE$

end

Event control5.sd_resume_call ⟨ordinary⟩ $\widehat{=}$
extends control5.sd_resume_call

when
grd1: $prog2 = null$
grd2: $sd\_suspend = TRUE$

then
act1: $prog2 := call\_sd\_resume$

end

Event control5·basal_start_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_start_return

when
grd3: $prog2 = return\_basal\_start$

then
control5.act4: $t\_basal := time + btime$
control5.act1: $basal\_work := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act3: $prog2 := null$

end

Event control5·basal_start_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_start_call

when
grd1: $basal\_work = FALSE$
grd2: $basal\_suspend = FALSE$
grd5: $prog2 = null$

then
act2: $par\_basal\_start\_t := timemodc$
act4: $prog2 := call\_basal\_start$

end

Event control5·basal_stop_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_stop_return

when
grd3: $prog2 = return\_basal\_stop$

then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_work := FALSE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act3: $prog2 := null$

end

Event control5·basal_stop_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_stop_call

when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd5: $prog2 = null$

then
act3: $prog2 := call\_basal\_stop$

end

Event control5·basal_suspend_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_suspend_return

when
grd3: $prog2 = return\_basal\_suspend$
then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_suspend := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_suspend_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_suspend_call
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

grd4: $basal\_rate \neq 0$
grd6: $prog2 = null$
then
act3: $prog2 := call\_basal\_suspend$
end
Event control5·basal_resume_return ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_resume_return
when
grd3: $prog2 = return\_basal\_resume$
then
control5.act1: $basal\_suspend := FALSE$
control5.act4: $t\_basal := btime + time$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_resume_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_resume_call
when
grd1: $basal\_suspend = TRUE$
grd4: $prog2 = null$
then
act2: $par\_basal\_resume\_t := timemodc$
act4: $prog2 := call\_basal\_resume$
end
Event control5·basal_update_rate_return ⟨ordinary⟩ $\widehat{=}$
extends control5 ·basal_update_rate_return
when
grd3: $prog2 = return\_basal\_update$
then
control5.act3: $t\_basal := time + btime$
control5.act1: $basal\_rate := basal\_rate\_in$
control5.act2: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_update_rate_call ⟨ordinary⟩ $\widehat{=}$
extends control5·basal_update_rate_call
when
grd1: $t\_basal = time$
grd2: $basal\_suspend = FALSE$
grd3: $basal\_work = TRUE$
grd5: $t\_basal \in dom(rate\_setting2 \rhd \{-1\})$

grd7: $prog2 = null$

then

act2: $par\_basal\_update\_rate\_t := t\_basal$

act4: $prog2 := call\_basal\_update$

end

Event control5.timer ⟨ordinary⟩ $\widehat{=}$

extends control5.timer

when

grd1:

$\neg($

$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$

$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$

$(time = t\_sd)) \lor$

$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$

$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$

$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$

$)$

grd2: $sdp\_add \neq 1$

then

act1: $time := time + 1$

end

Event NormalBolus.normal_bolus_start_calculate_insulin_needed ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_start_calculate_insulin_needed

any

$insulin$

where

NormalBolus.grd1: $insulin > 0$

NormalBolus.grd3: $normal\_add = 0$

grd2: $prog2 = call\_normal\_start$

then

NormalBolus.act1: $insulin\_needed := insulin$

NormalBolus.act2: $normal\_add := 1$

end

Event NormalBolus.normal_bolus_start_calculate_lasting_time ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_start_calculate_lasting_time

when

NormalBolus.grd1: $normal\_add = 1$

then

NormalBolus.act1: $normal\_delivering\_time := insulin\_needed / normal\_bolus\_rate$

NormalBolus.act2: $insulin\_needed := 0$

NormalBolus.act3: $normal\_add := 2$

end

Event NormalBolus.normal_bolus_delivery ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_delivery

when

NormalBolus.grd2: $normal\_add = 2$

then

NormalBolus.act1: $normal\_delivering\_rate := normal\_bolus\_rate$

NormalBolus.act2: $normal\_add := 3$

act2: $prog2 := return\_normal\_start$

act3: $normal\_delivering\_rate\_c := \lambda t \cdot t \in nb\_now..nb\_now + normal\_delivering\_time | normal\_bolus\_rate$

act5: $nb\_new\_now := nb\_now + normal\_delivering\_time$

end

Event NormalBolus.normal_bolus_suspend ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_suspend

any

ta

where

grd2: $prog2 = call\_normal\_suspend$

grd7: $ta \in nb\_now \mathinner{..} nb\_new\_now$

then

NormalBolus.act1: $normal\_delivering\_rate := 0$

NormalBolus.act2: $normal\_delivering\_time := 0$

NormalBolus.act3: $normal\_add := 0$

NormalBolus.act4: $normal\_bolus\_suspend := TRUE$

act2: $prog2 := return\_normal\_suspend$

act6: $normal\_delivering\_rate\_c := \lambda t \cdot t \geq ta | 0$

act5: $nb\_now := ta$

end

Event NormalBolus.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_finish

any

ta

where

grd2: $prog2 = call\_normal\_finish$

grd5: $ta = nb\_new\_now$

then

NormalBolus.act1: $normal\_delivering\_rate := 0$

NormalBolus.act2: $normal\_delivering\_time := 0$

NormalBolus.act3: $normal\_add := 0$

act2: $prog2 := return\_normal\_finish$

act4: $normal\_delivering\_rate\_c := \lambda t \cdot t \geq ta | 0$

act5: $nb\_now := ta$

end

Event NormalBolus.normal_bolus_resume ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_resume

any

ta

where

grd2: $prog2 = call\_normal\_resume$

grd3: $ta \geq nb\_now$

then

NormalBolus.act1: $normal\_bolus\_suspend := FALSE$

act2: $prog2 := return\_normal\_resume$

act3: $normal\_delivering\_rate := 0$

act4: $nb\_now := ta$

act5: $normal\_delivering\_rate\_c := \lambda t \cdot t \geq ta | 0$

end

Event Square_Dual_bolus2.start ⟨ordinary⟩ $\widehat{=}$

extends Square_Dual_bolus2.start

any

t

r

ctime

where

Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$

Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$

grd2: $prog2 = call\_sd\_start\_s$

grd4: $ctime \geq sd\_now$

then

Square_Dual_bolus2.act1: $state := deliver$

Square_Dual_bolus2.act2: $s\_r := r$

Square_Dual_bolus2.act3: $s\_t := t$

Square_Dual_bolus2.act6: $sd\_module := s$

Square_Dual_bolus2.act7: $d\_deliver\_time := t$
Square_Dual_bolus2.act8: $d\_deliver\_rate := r$
act2: $prog2 := return\_sd\_start\_s$
act9: $sd\_rate\_c := \lambda x \cdot x \in ctime \mathinner{..} ctime + t | r$
act10: $sd\_now := ctime$
act11: $sd\_new\_now := ctime + t$

end

Event Square_Dual_bolus2.start_dual ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.start_dual

any
t
r
td
ctime

where
Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$
Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$
Square_Dual_bolus2.grd4: $td \in \mathbb{N}_1$
grd1: $prog2 = call\_sd\_start\_d$
grd5: $ctime \geq sd\_now$

then
Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act2: $s\_r := r$
Square_Dual_bolus2.act3: $s\_t := t$
Square_Dual_bolus2.act6: $d\_deliver\_time := t + td$
Square_Dual_bolus2.act7: $d\_deliver\_rate := normal\_bolus\_rate$
Square_Dual_bolus2.act8: $d\_t := td$
Square_Dual_bolus2.act9: $sd\_module := d$
act1: $prog2 := return\_sd\_start\_d$
act11: $sd\_rate\_c := \lambda x \cdot x \in ctime \mathinner{..} ctime + td | normal\_bolus\_rate$
act12: $sd\_now := ctime$
act10: $sd\_new\_now := ctime + td$

end

Event Square_Dual_bolus2.update_to_dual ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.update_to_dual

any
ctime

where
grd1: $prog2 = call\_sd\_update$
grd5: $ctime = sd\_new\_now$

then
Square_Dual_bolus2.act2: $d\_deliver\_rate := s\_r$
Square_Dual_bolus2.act3: $sd\_flag := s$
act1: $prog2 := return\_sd\_update$
act4: $sd\_now := ctime$
act5: $sd\_new\_now := ctime + s\_t$
act6: $sd\_rate\_c := \lambda x \cdot x \in ctime \mathinner{..} ctime + s\_t | s\_r$

end

Event Square_Dual_bolus2.finish ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.finish

any
ctime

where
grd1: $prog2 = call\_sd\_finish$
grd3: $ctime = sd\_new\_now$

then
Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act4: $s\_r := 0$

Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_finish$
act10: $sd\_rate\_c := \lambda x \cdot x \geq ctime | 0$
act11: $sd\_now := ctime$

end

Event Square_Dual_bolus2.suspend ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.suspend

any
ctime
where
grd1: $prog2 = call\_sd\_suspend$
grd2: $ctime \in sd\_now \mathinner{..} sd\_new\_now$
then
Square_Dual_bolus2.act1: $state := suspend$
Square_Dual_bolus2.act4: $s\_r := 0$
Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_suspend$
act10: $sd\_now := ctime$
act11: $sd\_rate\_c := \lambda x \cdot x \geq ctime | 0$
end

Event Square_Dual_bolus2.resume ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.resume

any
ctime
where
grd1: $prog2 = call\_sd\_resume$
grd2: $ctime > sd\_now$
then
Square_Dual_bolus2.act1: $state := off$
act1: $prog2 := return\_sd\_resume$
act2: $sd\_now := ctime$
act3: $sd\_rate\_c := \lambda x \cdot x \geq ctime | 0$
end

Event Square_Dual_bolus2.preempted ⟨ordinary⟩ $\widehat{=}$
refines Square_Dual_bolus2.preempted

any
ctime
where
grd1: $prog2 = call\_sd\_preempt$
grd3: $ctime \in sd\_now \mathinner{..} sd\_new\_now$
grd4: $sd\_module = d \wedge sd\_flag = d \Rightarrow par\_sd\_preempt\_t \in s\_t + 1 \mathinner{..} d\_deliver\_time$
grd5: $sd\_module = d \wedge sd\_flag = s \Rightarrow par\_sd\_preempt\_t \in 0 \mathinner{..} s\_t$
then
Square_Dual_bolus2.act1: $state := preempt$
Square_Dual_bolus2.act4: $d\_deliver\_time := par\_sd\_preempt\_t$
Square_Dual_bolus2.act5: $d\_deliver\_rate := 0$
act1: $prog2 := return\_sd\_preempt$
act6: $sd\_now := ctime$
act7: $sd\_rate\_c := \lambda x \cdot x \geq ctime | 0$
end

Event Square_Dual_bolus2.resume_from_preempt ⟨ordinary⟩ $\widehat{=}$
extends Square_Dual_bolus2.resume_from_preempt
any
r
ctime
t2
where
Square_Dual_bolus2.grd1: $state = preempt$
Square_Dual_bolus2.grd2: $sd\_module = s \Rightarrow r = s\_r$
Square_Dual_bolus2.grd3: $sd\_module = d \wedge sd\_flag = d \Rightarrow r = normal\_bolus\_rate$
Square_Dual_bolus2.grd4: $sd\_module = d \wedge sd\_flag = s \Rightarrow r = s\_r$
grd1: $prog2 = call\_sd\_resume\_preempt$
grd5: $ctime > sd\_now$
grd6: $sd\_module = s \Rightarrow t2 = ctime + d\_deliver\_time$
grd7: $sd\_module = d \wedge sd\_flag = d \Rightarrow t2 = ctime + d\_deliver\_time - s\_t$
grd8: $sd\_module = d \wedge sd\_flag = s \Rightarrow t2 = ctime + d\_deliver\_time$
then
Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act4: $d\_deliver\_rate := r$
act1: $prog2 := return\_sd\_resume\_preempt$
act5: $sd\_now := ctime$
act6: $sd\_rate\_c := \lambda x \cdot x \in ctime \mathinner{..} t2 | r$
act7: $sd\_new\_now := t2$
end
Event Basal6·basal_suspend ⟨ordinary⟩ $\widehat{=}$
extends Basal6·basal_suspend
any
t
where
Basal6.grd3: $prog\_basal = null$
grd3: $prog2 = call\_basal\_suspend$
Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in \neq 0$
Basal6.grd2: ⟨theorem⟩ $basal\_mode = delivering$
grd4: $t \in fbegin \mathinner{..} fend$
then
Basal6.act1: $basal\_rate\_in := 0$
Basal6.act2: $basal\_mode := suspended$
act3: $prog2 := return\_basal\_suspend$
act4: $rate\_basal\_c := \lambda x \cdot x \geq t | 0$
act5: $fbegin := t$
end
Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6·change_setting
any
t
r
where
Basal6.grd5: $prog\_basal = null$
Basal6.grd6: $t \in 0 \mathinner{..} c - 1$
Basal6.grd7: $rate\_setting2(t) \neq -1$
Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$
then
Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$
end
Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$
extends Basal6.delete_setting
any
t

where

Basal6.grd5: $prog\_basal = null$

Basal6.grd2: $basal\_mode \neq suspended$

Basal6.grd6: $t \in 1 \mathinner{..} c - 1$

Basal6.grd7: $rate\_setting2(t) \neq -1$

grd1: $t \neq par\_basal\_update\_rate\_t$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd \{t \mapsto -1\}$

end

Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.add_setting

any

t

r

where

Basal6.grd9: $prog\_basal = null$

Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$

Basal6.grd4: $basal\_mode \neq suspended$

Basal6.grd5: $t \in 0 \mathinner{..} c - 1$

Basal6.grd6: $rate\_setting2(t) = -1$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd \{t \mapsto r\}$

end

Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_return

when

Basal6.grd8: $prog\_basal = return\_get\_max$

Basal6.grd9: $add\_resume = 2$

grd1: $prog2 = call\_basal\_resume$

then

Basal6.act1: $basal\_rate\_in := max\_value$

Basal6.act2: $basal\_mode := delivering$

Basal6.act3: $btime := min\_value - par\_get\_t$

Basal6.act4: $prog\_basal := null$

Basal6.act5: $add\_resume := 0$

act3: $prog2 := return\_basal\_resume$

act6: $rate\_basal\_c := \lambda x \cdot x \in par\_get\_t \mathinner{..} min\_value | max\_value$

act7: $fbegin := par\_get\_t$

act8: $fend := min\_value$

end

Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call

when

Basal6.grd6: $add\_resume = 0$

Basal6.grd5: $prog\_basal = null$

grd3: $prog2 = call\_basal\_resume$

Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in = 0$

Basal6.grd3: ⟨theorem⟩ $basal\_mode = suspended$

then

Basal6.act1: $par\_get\_t := par\_basal\_resume\_t$

Basal6.act2: $prog\_basal := call\_get\_min$

Basal6.act3: $add\_resume := 1$

end

Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$

Basal6.grd2: $add\_resume = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$

Basal6.act2: $add\_resume := 2$

end

Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6·rate_update_return

when

Basal6.grd12: $add\_update = 1$

Basal6.grd4: $prog\_basal = return\_get\_min$

grd1: $prog2 = call\_basal\_update$

then

Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$

Basal6.act2: $btime := min\_value - par\_get\_t$

Basal6.act3: $add\_update := 0$

act4: $prog\_basal := null$

act7: $prog2 := return\_basal\_update$

act8: $fbegin := par\_get\_t$

act5: $fend := min\_value$

act6: $rate\_basal\_c := \lambda x \cdot x \in par\_get\_t \mathinner{..} min\_value | rate\_setting2(par\_get\_t)$

end

Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.rate_update_call

when

Basal6.grd3: $add\_update = 0$

Basal6.grd2: $prog\_basal = null$

grd3: $prog2 = call\_basal\_update$

Basal6.grd5: ⟨theorem⟩ $basal\_mode = delivering$

Basal6.grd7: ⟨theorem⟩ $rate\_setting2(par\_basal\_update\_rate\_t) \neq -1$

then

Basal6.act1: $par\_get\_t := par\_basal\_update\_rate\_t$

Basal6.act2: $prog\_basal := call\_get\_min$

Basal6.act3: $add\_update := 1$

end

Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_return

when

Basal6.grd8: $add\_start = 2$

Basal6.grd9: $prog\_basal = return\_get\_max$

grd1: $prog2 = call\_basal\_start$

then

Basal6.act1: $basal\_mode := delivering$

Basal6.act2: $basal\_rate\_in := max\_value$

Basal6.act3: $btime := min\_value - par\_get\_t$

Basal6.act4: $add\_start := 0$

act4: $prog\_basal := null$

act7: $prog2 := return\_basal\_start$

act8: $fbegin := par\_get\_t$

act6: $fend := min\_value$

act5: $rate\_basal\_c := \lambda x \cdot x \in par\_get\_t \mathinner{..} min\_value | max\_value$

end

Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.start_call

when

Basal6.grd3: $add\_start = 0$

Basal6.grd2: $prog\_basal = null$

grd3: $prog2 = call\_basal\_start$

Basal6.grd4: ⟨theorem⟩ $basal\_mode = stop$

then

Basal6.act1: $par\_get\_t := par\_basal\_start\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_start := 1$

end

Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_start = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_start := 2$

end

Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$
extends Basal6.stop

any

t

where

Basal6.grd2: $prog\_basal = null$
Basal6.grd1: $basal\_mode = delivering$
grd3: $prog2 = call\_basal\_stop$
grd4: $t \in fbegin \mathinner{..} fend$

then

Basal6.act1: $basal\_mode := stop$
Basal6.act2: $basal\_rate\_in := 0$
act3: $prog2 := return\_basal\_stop$
act5: $fbegin := t$
act4: $rate\_basal\_c := \lambda x \cdot x \geq t | 0$

end

Event Basal6·get_min_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6·get_min_value_1

when

Basal6.grd4: $get\_min\_value\_add = 2$
Basal6.grd5: $par\_t = c$

then

Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$

end

Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_2

when

Basal6.grd5: $get\_min\_value\_add = 3$

then

Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$

end

Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_start

when

Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$

then

Basal6.act1: $par\_t := par\_get\_t + 1$
Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$

end
Event Basal6.find_min_value ⟨ordinary⟩ ≙
extends Basal6.find_min_value
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$
then
Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$
end
Event Basal6.find_min_value_2 ⟨ordinary⟩ ≙
extends Basal6.find_min_value_2
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$
then
Basal6.act1: $temp\_min := par\_t$
Basal6.act2: $get\_min\_value\_add := 3$
end
Event Basal6.get_max_value ⟨ordinary⟩ ≙
extends Basal6.get_max_value
when
Basal6.grd2: $get\_max\_value\_add = 2$
then
Basal6.act3: $prog\_basal := return\_get\_max$
Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$
Basal6.act2: $get\_max\_value\_add := 0$
end
Event Basal6.get_max_value_start ⟨ordinary⟩ ≙
extends Basal6.get_max_value_start
when
Basal6.grd2: $get\_max\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_max$
then
Basal6.act1: $get\_max\_start\_t := par\_get\_t$
Basal6.act2: $get\_max\_value\_add := 1$
Basal6.act3: $par\_t\_max := par\_get\_t$
end
Event Basal6·get_max_value_1 ⟨ordinary⟩ ≙
extends Basal6·get_max_value_1
when
Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd3: $par\_t\_max \geq 0$
Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$
then
Basal6.act1: $par\_t\_max := par\_t\_max - 1$
end
Event Basal6.get_max_value_2 ⟨ordinary⟩ ≙
extends Basal6.get_max_value_2
when
Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd2: $par\_t\_max \geq 0$
Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$
then
Basal6.act1: $get\_max\_value\_add := 2$
end
END

MACHINE control_Basal6_NormalBolus_2_Square_Dual_bolus2_5_c_2
REFINES control_Basal6_NormalBolus_2_Square_Dual_bolus2_5_c
SEES c_normalbolus_anim,c_prog2_anim,c_basal_anim,c_sd_bolus
VARIABLES

rate_setting2
normal_bolus_work
sd_preempted_by_normal
sd_bolus_work
sd_suspend
normal_suspend
basal_work
basal_suspend
pump_rate
basal_rate
normal_rate
sd_rate
time
t_basal
t_normal
t_sd
basal_rate_in
basal_mode
btime
par_basal_start_t
par_basal_resume_t
par_basal_update_rate_t
insulin_needed
normal_add
normal_delivering_time
normal_delivering_rate
normal_bolus_suspend
dmodule
d_update_time
state
s_r
s_t
d_deliver_time
d_deliver_rate
d_t
sd_module
sd_flag
prog2
par_sd_preempt_t
sdp_add
min_value
get_min_value_add
par_t
temp_min
get_min_start_t
max_value
get_max_start_t
get_max_value_add

par_t_max
prog_basal
par_get_t
add_resume
add_update
add_start
fbegin
fend
rate_basal_c
normal_delivering_rate_c
nb_now
nb_new_now
sd_now
sd_new_now
sd_rate_c

EVENTS

Initialisation ⟨extended⟩

begin

act1: $normal\_bolus\_work := FALSE$
act2: $sd\_bolus\_work := FALSE$
act3: $sd\_preempted\_by\_normal := FALSE$
act7: $sd\_suspend := FALSE$
act8: $normal\_suspend := FALSE$
act9: $basal\_work := FALSE$
act10: $basal\_suspend := FALSE$
act11: $pump\_rate := 0$
act12: $basal\_rate := 0$
act13: $normal\_rate := 0$
act14: $sd\_rate := 0$
act15: $time := 0$
act16: $t\_basal := 0$
act17: $t\_normal := 0$
act18: $t\_sd := 0$
act19: $dmodule := FALSE$
act20: $d\_update\_time := 0$
Basal1.act4: $btime := c$
Basal1.act2: $basal\_rate\_in := 0$
Basal1.act3: $basal\_mode := stop$
Basal6.act15: $prog\_basal := null$
Basal6.act16: $par\_get\_t := 0$
Basal6.act17: $add\_resume := 0$
Basal6.act18: $add\_update := 0$
Basal6.act19: $add\_start := 0$
Basal6.act5: $rate\_setting2 := (1 \mathinner{..} c - 1 \times \{-1\}) \cup \{0 \mapsto 0\}$
Basal6.act6: $min\_value := 0$
Basal6.act7: $max\_value := 0$
Basal6.act11: $get\_min\_value\_add := 0$
Basal6.act8: $par\_t := 0$
Basal6.act9: $temp\_min := 0$
Basal6.act10: $get\_min\_start\_t := 0$
Basal6.act12: $get\_max\_start\_t := 0$
Basal6.act13: $get\_max\_value\_add := 0$
Basal6.act14: $par\_t\_max := 0$
act22: $par\_basal\_start\_t := 0$
act23: $par\_basal\_resume\_t := 0$
act24: $par\_basal\_update\_rate\_t := 0$
NormalBolus.act1: $insulin\_needed := 0$

NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_delivering\_rate := 0$
NormalBolus.act4: $normal\_add := 0$
NormalBolus.act5: $normal\_bolus\_suspend := FALSE$
Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act2: $s\_r := 0$
Square_Dual_bolus2.act3: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_module := s$
Square_Dual_bolus2.act10: $sd\_flag := d$
act32: $prog2 := null$
act33: $par\_sd\_preempt\_t := 0$
act34: $sdp\_add := 0$
act35: $fbegin := 0$
act36: $fend := 0$
act37: $rate\_basal\_c := \{0 \mapsto 0\}$
act38: $normal\_delivering\_rate\_c := \{0 \mapsto 0\}$
act39: $nb\_now := 0$
act40: $nb\_new\_now := 0$
act41: $sd\_now := 0$
act42: $sd\_new\_now := 0$
act43: $sd\_rate\_c := \{0 \mapsto 0\}$

end

Event control5·normal_bolus_start_1_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_1_return

any

$t2$

where

grd6: $normal\_bolus\_work = FALSE$
grd7: $dmodule = TRUE \Rightarrow t2 = d\_update\_time - time \wedge time \neq d\_update\_time$
control5.grd2: $sd\_bolus\_work = TRUE$
control5.grd3: $sd\_preempted\_by\_normal = FALSE$
control5.grd4: $sd\_suspend = FALSE$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$
grd9: $prog2 = return\_normal\_start$
grd4: ⟨theorem⟩ $sdp\_add = 1$

then

control5.act6: $t\_normal := time + normal\_delivering\_time$
control5.act7: $t\_sd := t\_sd - time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $sd\_preempted\_by\_normal := TRUE$
control5.act3: $normal\_rate := normal\_delivering\_rate$
control5.act4: $sd\_rate := 0$
control5.act5: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act8: $d\_update\_time := t2$
act2: $prog2 := null$
act3: $sdp\_add := 0$

end

Event control5·normal_bolus_start_1_call_2 ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_1_call_2

when

grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_preempt$
grd8: ⟨theorem⟩ $sdp\_add = 1$

then
act2: $prog2 := call\_normal\_start$
end
Event control5·normal_bolus_start_1_call_sd_preempt ⟨ordinary⟩ $\widehat{=}$
extends control5 ·normal_bolus_start_1_call_sd_preempt
when
grd1: $prog2 = null$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $sd\_suspend = FALSE$
grd6: $dmodule = TRUE \Rightarrow time \neq d\_update\_time$
grd7: $sdp\_add = 0$
then
act1: $prog2 := call\_sd\_preempt$
act2: $par\_sd\_preempt\_t := t\_sd - time$
act3: $sdp\_add := 1$
end
Event control5.normal_bolus_start_2_return ⟨ordinary⟩ $\widehat{=}$
extends control5.normal_bolus_start_2_return
when
control5.grd2: $sd\_bolus\_work = FALSE$
grd2: $prog2 = return\_normal\_start$
then
control5·act4: $t\_normal := time + normal\_delivering\_time$
control5.act1: $normal\_bolus\_work := TRUE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := normal\_delivering\_rate + basal\_rate$
act2: $prog2 := null$
end
Event control5·normal_bolus_start_2_call ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_start_2_call
when
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_bolus\_work = FALSE$
grd4: $normal\_suspend = FALSE$
grd5: $prog2 = null$
then
act2: $prog2 := call\_normal\_start$
end
Event control5·normal_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$
extends control5·normal_bolus_finish_return
when
grd2: $prog2 = return\_normal\_finish$
then
control5.act4: $t\_normal := 0$
control5.act1: $normal\_bolus\_work := FALSE$
control5.act2: $normal\_rate := normal\_delivering\_rate$
control5.act3: $pump\_rate := basal\_rate$
act2: $prog2 := null$
end
Event control5·normal_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$
extends control5 ·normal_bolus_finish_call
when
grd2: $time = t\_normal$
grd3: $normal\_bolus\_work = TRUE$
grd4: $normal\_suspend = FALSE$

grd5: $prog2 = null$

then

act2: $prog2 := call\_normal\_finish$

end

Event control5·normal_suspend_return ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_suspend_return

when

grd2: $prog2 = return\_normal\_suspend$

then

control5.act4: $t\_normal := t\_normal - time$

control5.act1: $normal\_suspend := TRUE$

control5.act2: $normal\_rate := normal\_delivering\_rate$

control5.act3: $pump\_rate := basal\_rate$

act3: $prog2 := null$

end

Event control5·normal_bolus_suspend_call ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_bolus_suspend_call

when

grd2: $normal\_bolus\_work = TRUE$

grd3: $normal\_suspend = FALSE$

grd4: $prog2 = null$

then

act2: $prog2 := call\_normal\_suspend$

end

Event control5·normal_resume_return ⟨ordinary⟩ $\widehat{=}$

extends control5·normal_resume_return

when

grd2: $prog2 = return\_normal\_resume$

then

control5.act4: $t\_normal := 0$

control5.act1: $normal\_suspend := FALSE$

control5.act2: $normal\_rate := 0$

control5.act3: $pump\_rate := basal\_rate$

act2: $normal\_bolus\_work := FALSE$

act3: $prog2 := null$

end

Event control5·normal_bolus_resume_call ⟨ordinary⟩ $\widehat{=}$

extends control5 ·normal_bolus_resume_call

when

grd2: $normal\_suspend = TRUE$

grd3: $prog2 = null$

then

act2: $prog2 := call\_normal\_resume$

end

Event control5·square_or_dual_bolus_start_s_return ⟨ordinary⟩ $\widehat{=}$

extends control5 ·square_or_dual_bolus_start_s_return

when

grd1: $sd\_bolus\_work = FALSE$

grd2: $normal\_bolus\_work = FALSE$

grd3: $sd\_suspend = FALSE$

grd6: $prog2 = return\_sd\_start\_s$

then

act1: $sd\_bolus\_work := TRUE$

act2: $sd\_rate := d\_deliver\_rate$

act3: $pump\_rate := d\_deliver\_rate + basal\_rate$

act4: $t\_sd := time + d\_deliver\_time$

act5: $dmodule := FALSE$

act6: $prog2 := null$

end

Event control5square_or_dual_bolus_start_s_call ⟨ordinary⟩ $\widehat{=}$

extends control5square_or_dual_bolus_start_s_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_start\_s$

end

Event control5·square_or_dual_bolus_start_d_return ⟨ordinary⟩ $\widehat{=}$

extends control5 ·square_or_dual_bolus_start_d_return

when

grd1: $sd\_bolus\_work = FALSE$
grd2: $normal\_bolus\_work = FALSE$
grd3: $sd\_suspend = FALSE$
grd8: $prog2 = return\_sd\_start\_d$

then

act1: $sd\_bolus\_work := TRUE$
act5: $dmodule := TRUE$
act6: $d\_update\_time := time + d\_t$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + d\_deliver\_time$
act7: $prog2 := null$

end

Event control5·square_or_dual_bolus_start_d_call ⟨ordinary⟩ $\widehat{=}$

extends control5·square_or_dual_bolus_start_d_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = FALSE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_start\_d$

end

Event control5.square_or_dual_bolus_finish_return ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_bolus_finish_return

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$
grd6: $prog2 = return\_sd\_finish$

then

act1: $sd\_bolus\_work := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := d\_deliver\_time$
act5: $dmodule := FALSE$
act6: $prog2 := null$

end

Event control5.square_or_dual_bolus_finish_call ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_bolus_finish_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $time = t\_sd$

then

act1: $prog2 := call\_sd\_finish$

end

Event control5.square_or_dual_bolus_resume_from_normal_return ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_bolus_resume_from_normal_return

any

$t2$

where

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = TRUE$
grd3: $normal\_bolus\_work = FALSE$
grd4: $sd\_suspend = FALSE$
grd6: $prog2 = return\_sd\_resume\_preempt$
grd7: $dmodule = TRUE \Rightarrow t2 = time + d\_update\_time$
grd8: $dmodule = FALSE \Rightarrow t2 = 0$

then

act1: $sd\_preempted\_by\_normal := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := d\_deliver\_rate + basal\_rate$
act4: $t\_sd := time + t\_sd$
act5: $prog2 := null$
act6: $d\_update\_time := t2$

end

Event control5.square_or_dual_bolus_resume_from_normal_call ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_bolus_resume_from_normal_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = TRUE$
grd4: $normal\_bolus\_work = FALSE$
grd5: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_resume\_preempt$

end

Event control5.sd_suspend_return ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_suspend_return

when

grd1: $sd\_bolus\_work = TRUE$
grd2: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$
grd5: $prog2 = return\_sd\_suspend$

then

act1: $sd\_suspend := TRUE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := t\_sd - time$
act5: $dmodule := FALSE$
act6: $prog2 := null$
act7: $d\_update\_time := 0$

end

Event control5.sd_suspend_call ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_suspend_call

when

grd1: $prog2 = null$
grd2: $sd\_bolus\_work = TRUE$
grd3: $sd\_preempted\_by\_normal = FALSE$
grd4: $sd\_suspend = FALSE$

then

act1: $prog2 := call\_sd\_suspend$

end

Event control5.square_or_dual_update_rate_return ⟨ordinary⟩ $\widehat{=}$

extends control5.square_or_dual_update_rate_return

when

grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd6: $dmodule = TRUE$
grd7: $time = d\_update\_time$
grd8: $prog2 = return\_sd\_update$

then

act1: $sd\_rate := d\_deliver\_rate$
act2: $pump\_rate := d\_deliver\_rate + basal\_rate$
act3: $dmodule := FALSE$
act4: $prog2 := null$
act5: $d\_update\_time := 0$

end

Event control5·square_or_dual_update_rate_call ⟨ordinary⟩ $\widehat{=}$

extends control5·square_or_dual_update_rate_call

when

grd1: $prog2 = null$
grd2: $sd\_suspend = FALSE$
grd3: $sd\_bolus\_work = TRUE$
grd4: $sd\_preempted\_by\_normal = FALSE$
grd5: $dmodule = TRUE$
grd6: $time = d\_update\_time$

then

act1: $prog2 := call\_sd\_update$

end

Event control5.sd_resume_return ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_resume_return

when

grd1: $sd\_suspend = TRUE$
grd3: $prog2 = return\_sd\_resume$

then

act1: $sd\_suspend := FALSE$
act2: $sd\_rate := d\_deliver\_rate$
act3: $pump\_rate := basal\_rate$
act4: $t\_sd := 0$
act5: $prog2 := null$
act6: $sd\_bolus\_work := FALSE$

end

Event control5.sd_resume_call ⟨ordinary⟩ $\widehat{=}$

extends control5.sd_resume_call

when

grd1: $prog2 = null$
grd2: $sd\_suspend = TRUE$

then

act1: $prog2 := call\_sd\_resume$

end

Event control5·basal_start_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_start_return

when
grd3: $prog2 = return\_basal\_start$
then
control5.act4: $t\_basal := time + btime$
control5.act1: $basal\_work := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_start_call ⟨ordinary⟩ ≙
extends control5·basal_start_call
when
grd1: $basal\_work = FALSE$
grd2: $basal\_suspend = FALSE$
grd5: $prog2 = null$
then
act2: $par\_basal\_start\_t := timemodc$
act4: $prog2 := call\_basal\_start$
end
Event control5·basal_stop_return ⟨ordinary⟩ ≙
extends control5·basal_stop_return
when
grd3: $prog2 = return\_basal\_stop$
then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_work := FALSE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_stop_call ⟨ordinary⟩ ≙
extends control5·basal_stop_call
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$
grd5: $prog2 = null$
then
act3: $prog2 := call\_basal\_stop$
end
Event control5·basal_suspend_return ⟨ordinary⟩ ≙
extends control5·basal_suspend_return
when
grd3: $prog2 = return\_basal\_suspend$
then
control5.act4: $t\_basal := 0$
control5.act1: $basal\_suspend := TRUE$
control5.act2: $basal\_rate := basal\_rate\_in$
control5.act3: $pump\_rate := normal\_rate + sd\_rate$
act3: $prog2 := null$
end
Event control5·basal_suspend_call ⟨ordinary⟩ ≙
extends control5·basal_suspend_call
when
grd1: $basal\_work = TRUE$
grd2: $basal\_suspend = FALSE$

grd4: $basal\_rate \neq 0$

grd6: $prog2 = null$

then

act3: $prog2 := call\_basal\_suspend$

end

Event control5·basal_resume_return ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_resume_return

when

grd3: $prog2 = return\_basal\_resume$

then

control5.act1: $basal\_suspend := FALSE$

control5.act4: $t\_basal := btime + time$

control5.act2: $basal\_rate := basal\_rate\_in$

control5.act3: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$

act3: $prog2 := null$

end

Event control5·basal_resume_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_resume_call

when

grd1: $basal\_suspend = TRUE$

grd4: $prog2 = null$

then

act2: $par\_basal\_resume\_t := timemodc$

act4: $prog2 := call\_basal\_resume$

end

Event control5·basal_update_rate_return ⟨ordinary⟩ $\widehat{=}$

extends control5 ·basal_update_rate_return

when

grd3: $prog2 = return\_basal\_update$

then

control5.act3: $t\_basal := time + btime$

control5.act1: $basal\_rate := basal\_rate\_in$

control5.act2: $pump\_rate := basal\_rate\_in + normal\_rate + sd\_rate$

act3: $prog2 := null$

end

Event control5·basal_update_rate_call ⟨ordinary⟩ $\widehat{=}$

extends control5·basal_update_rate_call

when

grd1: $t\_basal = time$

grd2: $basal\_suspend = FALSE$

grd3: $basal\_work = TRUE$

grd5: $t\_basal \in dom(rate\_setting2 \rhd \{-1\})$

grd7: $prog2 = null$

then

act2: $par\_basal\_update\_rate\_t := t\_basal$

act4: $prog2 := call\_basal\_update$

end

Event control5.timer ⟨ordinary⟩ $\widehat{=}$

extends control5.timer

when

grd1:

$\neg($

$((normal\_bolus\_work = TRUE) \land (normal\_suspend = FALSE) \land (time = t\_normal)) \lor$

$((sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land (sd\_suspend = FALSE) \land$

$(time = t\_sd)) \lor$

$(((sd\_suspend = FALSE) \land (sd\_bolus\_work = TRUE) \land (sd\_preempted\_by\_normal = FALSE) \land$

$(dmodule = TRUE) \land (time = d\_update\_time))) \lor$

$(((basal\_suspend = FALSE) \land (basal\_work = TRUE) \land (t\_basal = time)))$

$)$

grd2: $sdp\_add \neq 1$

then

act1: $time := time + 1$

end

Event NormalBolus.normal_bolus_start_calculate_insulin_needed ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_start_calculate_insulin_needed

any

$insulin$

where

NormalBolus.grd1: $insulin > 0$

NormalBolus.grd3: $normal\_add = 0$

grd2: $prog2 = call\_normal\_start$

then

NormalBolus.act1: $insulin\_needed := insulin$

NormalBolus.act2: $normal\_add := 1$

end

Event NormalBolus.normal_bolus_start_calculate_lasting_time ⟨ordinary⟩ $\widehat{=}$

extends NormalBolus.normal_bolus_start_calculate_lasting_time

when

NormalBolus.grd1: $normal\_add = 1$

then

NormalBolus.act1: $normal\_delivering\_time := insulin\_needed / normal\_bolus\_rate$

NormalBolus.act2: $insulin\_needed := 0$

NormalBolus.act3: $normal\_add := 2$

end

Event NormalBolus.normal_bolus_delivery ⟨ordinary⟩ $\widehat{=}$

refines NormalBolus.normal_bolus_delivery

when

NormalBolus.grd2: $normal\_add = 2$

then

NormalBolus.act1: $normal\_delivering\_rate := normal\_bolus\_rate$

NormalBolus.act2: $normal\_add := 3$

act2: $prog2 := return\_normal\_start$

act3: $normal\_delivering\_rate\_c := \lambda t \cdot t \in nb\_now..nb\_now + normal\_delivering\_time | normal\_bolus\_rate$

act5: $nb\_new\_now := nb\_now + normal\_delivering\_time$

end

Event NormalBolus.normal_bolus_suspend ⟨ordinary⟩ $\widehat{=}$

refines NormalBolus.normal_bolus_suspend

when

grd2: $prog2 = call\_normal\_suspend$

grd7: $time \in nb\_now \,..\, nb\_new\_now$

with

ta: $ta = time$

then

NormalBolus.act1: $normal\_delivering\_rate := 0$

NormalBolus.act2: $normal\_delivering\_time := 0$

NormalBolus.act3: $normal\_add := 0$

NormalBolus.act4: $normal\_bolus\_suspend := TRUE$

act2: $prog2 := return\_normal\_suspend$

act6: $normal\_delivering\_rate\_c := \lambda t \cdot t \geq time | 0$

act5: $nb\_now := time$

end

Event NormalBolus.normal_bolus_finish ⟨ordinary⟩ $\widehat{=}$

refines NormalBolus.normal_bolus_finish

when

grd2: $prog2 = call\_normal\_finish$

grd5: $time = nb\_new\_now$
grd6: $t\_normal = time$

with

ta: $ta = time$

then

NormalBolus.act1: $normal\_delivering\_rate := 0$
NormalBolus.act2: $normal\_delivering\_time := 0$
NormalBolus.act3: $normal\_add := 0$
act2: $prog2 := return\_normal\_finish$
act4: $normal\_delivering\_rate\_c := \lambda t \cdot t \geq time | 0$
act5: $nb\_now := time$

end

Event NormalBolus.normal_bolus_resume ⟨ordinary⟩ $\widehat{=}$

refines NormalBolus.normal_bolus_resume

when

grd2: $prog2 = call\_normal\_resume$
grd3: $time \geq nb\_now$

with

ta: $ta = time$

then

NormalBolus.act1: $normal\_bolus\_suspend := FALSE$
act2: $prog2 := return\_normal\_resume$
act3: $normal\_delivering\_rate := 0$
act4: $nb\_now := time$
act5: $normal\_delivering\_rate\_c := \lambda t \cdot t \geq time | 0$

end

Event Square_Dual_bolus2.start ⟨ordinary⟩ $\widehat{=}$

refines Square_Dual_bolus2.start

any

t
r

where

Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$
Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$
grd2: $prog2 = call\_sd\_start\_s$
grd4: $time \geq sd\_now$

with

ctime: $ctime = time$

then

Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act2: $s\_r := r$
Square_Dual_bolus2.act3: $s\_t := t$
Square_Dual_bolus2.act6: $sd\_module := s$
Square_Dual_bolus2.act7: $d\_deliver\_time := t$
Square_Dual_bolus2.act8: $d\_deliver\_rate := r$
act2: $prog2 := return\_sd\_start\_s$
act9: $sd\_rate\_c := \lambda x \cdot x \in time \mathinner{..} time + t | r$
act10: $sd\_now := time$
act11: $sd\_new\_now := time + t$

end

Event Square_Dual_bolus2.start_dual ⟨ordinary⟩ $\widehat{=}$

refines Square_Dual_bolus2.start_dual

any

t
r
td

where

Square_Dual_bolus2.grd2: $t \in \mathbb{N}_1$

Square_Dual_bolus2.grd3: $r \in \mathbb{N}_1$
Square_Dual_bolus2.grd4: $td \in \mathbb{N}_1$
grd1: $prog2 = call\_sd\_start\_d$
grd5: $time \geq sd\_now$

with

ctime: $ctime = time$

then

Square_Dual_bolus2.act1: $state := deliver$
Square_Dual_bolus2.act2: $s\_r := r$
Square_Dual_bolus2.act3: $s\_t := t$
Square_Dual_bolus2.act6: $d\_deliver\_time := t + td$
Square_Dual_bolus2.act7: $d\_deliver\_rate := normal\_bolus\_rate$
Square_Dual_bolus2.act8: $d\_t := td$
Square_Dual_bolus2.act9: $sd\_module := d$
act1: $prog2 := return\_sd\_start\_d$
act11: $sd\_rate\_c := \lambda x \cdot x \in time \mathinner{..} time + td | normal\_bolus\_rate$
act12: $sd\_now := time$
act10: $sd\_new\_now := time + td$

end

Event Square_Dual_bolus2.update_to_dual ⟨ordinary⟩ $\widehat{=}$

refines Square_Dual_bolus2.update_to_dual

when

grd1: $prog2 = call\_sd\_update$
grd5: $time = sd\_new\_now$

with

ctime: $ctime = time$

then

Square_Dual_bolus2.act2: $d\_deliver\_rate := s\_r$
Square_Dual_bolus2.act3: $sd\_flag := s$
act1: $prog2 := return\_sd\_update$
act4: $sd\_now := time$
act5: $sd\_new\_now := time + s\_t$
act6: $sd\_rate\_c := \lambda x \cdot x \in time \mathinner{..} time + s\_t | s\_r$

end

Event Square_Dual_bolus2.finish ⟨ordinary⟩ $\widehat{=}$

refines Square_Dual_bolus2.finish

when

grd1: $prog2 = call\_sd\_finish$
grd3: $time = sd\_new\_now$

with

ctime: $ctime = time$

then

Square_Dual_bolus2.act1: $state := off$
Square_Dual_bolus2.act4: $s\_r := 0$
Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_finish$
act10: $sd\_rate\_c := \lambda x \cdot x \geq time | 0$
act11: $sd\_now := time$

end

Event Square_Dual_bolus2.suspend ⟨ordinary⟩ $\widehat{=}$

refines Square_Dual_bolus2.suspend

when

grd1: $prog2 = call\_sd\_suspend$
grd2: $time \in sd\_now \mathinner{..} sd\_new\_now$

with
ctime: $ctime = time$
then
Square_Dual_bolus2.act1: $state := suspend$
Square_Dual_bolus2.act4: $s\_r := 0$
Square_Dual_bolus2.act5: $s\_t := 0$
Square_Dual_bolus2.act6: $d\_deliver\_time := 0$
Square_Dual_bolus2.act7: $d\_deliver\_rate := 0$
Square_Dual_bolus2.act8: $d\_t := 0$
Square_Dual_bolus2.act9: $sd\_flag := d$
act1: $prog2 := return\_sd\_suspend$
act10: $sd\_now := time$
act11: $sd\_rate\_c := \lambda x \cdot x \geq time | 0$
end

Event Square_Dual_bolus2.resume ⟨ordinary⟩ $\widehat{=}$
refines Square_Dual_bolus2.resume
when
grd1: $prog2 = call\_sd\_resume$
grd2: $time > sd\_now$
with
ctime: $ctime = time$
then
Square_Dual_bolus2.act1: $state := off$
act1: $prog2 := return\_sd\_resume$
act2: $sd\_now := time$
act3: $sd\_rate\_c := \lambda x \cdot x \geq time | 0$
end

Event Square_Dual_bolus2.preempted ⟨ordinary⟩ $\widehat{=}$
refines Square_Dual_bolus2.preempted
when
grd1: $prog2 = call\_sd\_preempt$
grd3: $time \in sd\_now \mathinner{..} sd\_new\_now$
grd4: $sd\_module = d \wedge sd\_flag = d \Rightarrow par\_sd\_preempt\_t \in s\_t + 1 \mathinner{..} d\_deliver\_time$
grd5: $sd\_module = d \wedge sd\_flag = s \Rightarrow par\_sd\_preempt\_t \in 0 \mathinner{..} s\_t$
with
ctime: $ctime = time$
then
Square_Dual_bolus2.act1: $state := preempt$
Square_Dual_bolus2.act4: $d\_deliver\_time := par\_sd\_preempt\_t$
Square_Dual_bolus2.act5: $d\_deliver\_rate := 0$
act1: $prog2 := return\_sd\_preempt$
act6: $sd\_now := time$
act7: $sd\_rate\_c := \lambda x \cdot x \geq time | 0$
end

Event Square_Dual_bolus2.resume_from_preempt ⟨ordinary⟩ $\widehat{=}$
refines Square_Dual_bolus2.resume_from_preempt
any
r
t2
where
Square_Dual_bolus2.grd1: $state = preempt$
Square_Dual_bolus2.grd2: $sd\_module = s \Rightarrow r = s\_r$
Square_Dual_bolus2.grd3: $sd\_module = d \wedge sd\_flag = d \Rightarrow r = normal\_bolus\_rate$
Square_Dual_bolus2.grd4: $sd\_module = d \wedge sd\_flag = s \Rightarrow r = s\_r$
grd1: $prog2 = call\_sd\_resume\_preempt$
grd5: $time > sd\_now$
grd6: $sd\_module = s \Rightarrow t2 = time + d\_deliver\_time$
grd7: $sd\_module = d \wedge sd\_flag = d \Rightarrow t2 = time + d\_deliver\_time - s\_t$

grd8: $sd\_module = d \land sd\_flag = s \Rightarrow t2 = time + d\_deliver\_time$

with

ctime: $ctime = time$

then

Square_Dual_bolus2.act1: $state := deliver$

Square_Dual_bolus2.act4: $d\_deliver\_rate := r$

act1: $prog2 := return\_sd\_resume\_preempt$

act5: $sd\_now := time$

act6: $sd\_rate\_c := \lambda x \cdot x \in time \mathinner{..} t2 | r$

act7: $sd\_new\_now := t2$

end

Event Basal6·basal_suspend ⟨ordinary⟩ $\widehat{=}$

refines Basal6·basal_suspend

when

Basal6.grd3: $prog\_basal = null$

grd3: $prog2 = call\_basal\_suspend$

Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in \neq 0$

Basal6.grd2: ⟨theorem⟩ $basal\_mode = delivering$

grd4: $time \in fbegin \mathinner{..} fend$

with

t: $t = time$

then

Basal6.act1: $basal\_rate\_in := 0$

Basal6.act2: $basal\_mode := suspended$

act3: $prog2 := return\_basal\_suspend$

act4: $rate\_basal\_c := \lambda x \cdot x \geq time | 0$

act5: $fbegin := time$

end

Event Basal6·change_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6·change_setting

any

t

r

where

Basal6.grd5: $prog\_basal = null$

Basal6.grd6: $t \in 0 \mathinner{..} c - 1$

Basal6.grd7: $rate\_setting2(t) \neq -1$

Basal6.grd2: $r \in 0 \mathinner{..} basal\_max$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$

end

Event Basal6.delete_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.delete_setting

any

t

where

Basal6.grd5: $prog\_basal = null$

Basal6.grd2: $basal\_mode \neq suspended$

Basal6.grd6: $t \in 1 \mathinner{..} c - 1$

Basal6.grd7: $rate\_setting2(t) \neq -1$

grd1: $t \neq par\_basal\_update\_rate\_t$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto -1\}$

end

Event Basal6.add_setting ⟨ordinary⟩ $\widehat{=}$

extends Basal6.add_setting

any

t

r

where

Basal6.grd9: $prog\_basal = null$

Basal6.grd3: $r \in 0 \mathinner{..} basal\_max$

Basal6.grd4: $basal\_mode \neq suspended$

Basal6.grd5: $t \in 0 \mathinner{..} c - 1$

Basal6.grd6: $rate\_setting2(t) = -1$

then

Basal6.act2: $rate\_setting2 := rate\_setting2 \lhd\!\!+ \{t \mapsto r\}$

end

Event Basal6.basal_resume_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_return

when

Basal6.grd8: $prog\_basal = return\_get\_max$

Basal6.grd9: $add\_resume = 2$

grd1: $prog2 = call\_basal\_resume$

then

Basal6.act1: $basal\_rate\_in := max\_value$

Basal6.act2: $basal\_mode := delivering$

Basal6.act3: $btime := min\_value - par\_get\_t$

Basal6.act4: $prog\_basal := null$

Basal6.act5: $add\_resume := 0$

act3: $prog2 := return\_basal\_resume$

act6: $rate\_basal\_c := \lambda x \cdot x \in par\_get\_t \mathinner{..} min\_value | max\_value$

act7: $fbegin := par\_get\_t$

act8: $fend := min\_value$

end

Event Basal6.basal_resume_call ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call

when

Basal6.grd6: $add\_resume = 0$

Basal6.grd5: $prog\_basal = null$

grd3: $prog2 = call\_basal\_resume$

Basal6.grd1: ⟨theorem⟩ $basal\_rate\_in = 0$

Basal6.grd3: ⟨theorem⟩ $basal\_mode = suspended$

then

Basal6.act1: $par\_get\_t := par\_basal\_resume\_t$

Basal6.act2: $prog\_basal := call\_get\_min$

Basal6.act3: $add\_resume := 1$

end

Event Basal6.basal_resume_call_2 ⟨ordinary⟩ $\widehat{=}$

extends Basal6.basal_resume_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$

Basal6.grd2: $add\_resume = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$

Basal6.act2: $add\_resume := 2$

end

Event Basal6·rate_update_return ⟨ordinary⟩ $\widehat{=}$

extends Basal6·rate_update_return

when

Basal6.grd12: $add\_update = 1$

Basal6.grd4: $prog\_basal = return\_get\_min$

grd1: $prog2 = call\_basal\_update$

then

Basal6.act1: $basal\_rate\_in := rate\_setting2(par\_get\_t)$

Basal6.act2: $btime := min\_value - par\_get\_t$

Basal6.act3: $add\_update := 0$
act4: $prog\_basal := null$
act7: $prog2 := return\_basal\_update$
act8: $fbegin := par\_get\_t$
act5: $fend := min\_value$
act6: $rate\_basal\_c := \lambda x \cdot x \in par\_get\_t \mathinner{..} min\_value | rate\_setting2(par\_get\_t)$

end

Event Basal6.rate_update_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.rate_update_call

when

Basal6.grd3: $add\_update = 0$
Basal6.grd2: $prog\_basal = null$
grd3: $prog2 = call\_basal\_update$
Basal6.grd5: ⟨theorem⟩ $basal\_mode = delivering$
Basal6.grd7: ⟨theorem⟩ $rate\_setting2(par\_basal\_update\_rate\_t) \neq -1$

then

Basal6.act1: $par\_get\_t := par\_basal\_update\_rate\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_update := 1$

end

Event Basal6.start_return ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_return

when

Basal6.grd8: $add\_start = 2$
Basal6.grd9: $prog\_basal = return\_get\_max$
grd1: $prog2 = call\_basal\_start$

then

Basal6.act1: $basal\_mode := delivering$
Basal6.act2: $basal\_rate\_in := max\_value$
Basal6.act3: $btime := min\_value - par\_get\_t$
Basal6.act4: $add\_start := 0$
act4: $prog\_basal := null$
act7: $prog2 := return\_basal\_start$
act8: $fbegin := par\_get\_t$
act6: $fend := min\_value$
act5: $rate\_basal\_c := \lambda x \cdot x \in par\_get\_t \mathinner{..} min\_value | max\_value$

end

Event Basal6.start_call ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call

when

Basal6.grd3: $add\_start = 0$
Basal6.grd2: $prog\_basal = null$
grd3: $prog2 = call\_basal\_start$
Basal6.grd4: ⟨theorem⟩ $basal\_mode = stop$

then

Basal6.act1: $par\_get\_t := par\_basal\_start\_t$
Basal6.act2: $prog\_basal := call\_get\_min$
Basal6.act3: $add\_start := 1$

end

Event Basal6.start_call_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.start_call_2

when

Basal6.grd1: $prog\_basal = return\_get\_min$
Basal6.grd2: $add\_start = 1$

then

Basal6.act1: $prog\_basal := call\_get\_max$
Basal6.act2: $add\_start := 2$

end

Event Basal6.stop ⟨ordinary⟩ $\widehat{=}$
refines Basal6.stop
when
Basal6.grd2: $prog\_basal = null$
Basal6.grd1: $basal\_mode = delivering$
grd3: $prog2 = call\_basal\_stop$
grd4: $time \in fbegin \mathinner{..} fend$
with
t: $t = time$
then
Basal6.act1: $basal\_mode := stop$
Basal6.act2: $basal\_rate\_in := 0$
act3: $prog2 := return\_basal\_stop$
act5: $fbegin := time$
act4: $rate\_basal\_c := \lambda x \cdot x \geq time | 0$
end
Event Basal6·get_min_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6·get_min_value_1
when
Basal6.grd4: $get\_min\_value\_add = 2$
Basal6.grd5: $par\_t = c$
then
Basal6.act1: $min\_value := c$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_2
when
Basal6.grd5: $get\_min\_value\_add = 3$
then
Basal6.act1: $min\_value := temp\_min$
Basal6.act2: $get\_min\_value\_add := 0$
Basal6.act3: $prog\_basal := return\_get\_min$
end
Event Basal6.get_min_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_min_value_start
when
Basal6.grd2: $get\_min\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_min$
then
Basal6.act1: $par\_t := par\_get\_t + 1$
Basal6.act2: $get\_min\_value\_add := 1$
Basal6.act3: $get\_min\_start\_t := par\_get\_t$
end
Event Basal6.find_min_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value
when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) = -1$
then
Basal6.act1: $par\_t := par\_t + 1$
Basal6.act2: $get\_min\_value\_add := 2$
end
Event Basal6.find_min_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.find_min_value_2

when
Basal6.grd1: $par\_t < c$
Basal6.grd2: $get\_min\_value\_add = 1 \lor get\_min\_value\_add = 2$
Basal6.grd3: $rate\_setting2(par\_t) \neq -1$
then
Basal6.act1: $temp\_min := par\_t$
Basal6.act2: $get\_min\_value\_add := 3$
end
Event Basal6.get_max_value ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value
when
Basal6.grd2: $get\_max\_value\_add = 2$
then
Basal6.act3: $prog\_basal := return\_get\_max$
Basal6.act1: $max\_value := rate\_setting2(par\_t\_max)$
Basal6.act2: $get\_max\_value\_add := 0$
end
Event Basal6.get_max_value_start ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_start
when
Basal6.grd2: $get\_max\_value\_add = 0$
Basal6.grd3: $prog\_basal = call\_get\_max$
then
Basal6.act1: $get\_max\_start\_t := par\_get\_t$
Basal6.act2: $get\_max\_value\_add := 1$
Basal6.act3: $par\_t\_max := par\_get\_t$
end
Event Basal6·get_max_value_1 ⟨ordinary⟩ $\widehat{=}$
extends Basal6·get_max_value_1
when
Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd3: $par\_t\_max \geq 0$
Basal6.grd2: $rate\_setting2(par\_t\_max) = -1$
then
Basal6.act1: $par\_t\_max := par\_t\_max - 1$
end
Event Basal6.get_max_value_2 ⟨ordinary⟩ $\widehat{=}$
extends Basal6.get_max_value_2
when
Basal6.grd1: $get\_max\_value\_add = 1$
Basal6.grd2: $par\_t\_max \geq 0$
Basal6.grd3: $rate\_setting2(par\_t\_max) \neq -1$
then
Basal6.act1: $get\_max\_value\_add := 2$
end
END