

University of Southampton Research Repository ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

Appendix A

Models of the Controlled Water Tank System

Contents

A Models of the Controlled Water Tank System	1
A.1 Modelica Model of the Plant	3
A.2 Event-B Specification of the Controller	4
A.2.1 Context <code>ctx</code>	4
A.2.2 Abstract Machine <code>wtCtr0</code>	4
A.2.3 Concrete Machine <code>wtCtr1</code>	5

A.1 Modelica Model of the Plant

```
model ControlledWaterTankPlant
  "Physical aspect of the controlled water tank"
  inner Modelica.Fluid.System system;
  Modelica.Fluid.Vessels.OpenTank tank(
    nPorts=2,
    height=3,
    crossArea=1,
    redeclare package Medium =
      Modelica.Media.Water.ConstantPropertyLiquidWater,
    portsData={Modelica.Fluid.Vessels.BaseClasses.VesselPortsData(
      diameter=0.1,
      height=3),
      Modelica.Fluid.Vessels.BaseClasses.VesselPortsData(
        diameter=0.2)},
    level_start=0);
  Modelica.Fluid.Sources.FixedBoundary source(
    redeclare package Medium =
      Modelica.Media.Water.ConstantPropertyLiquidWater,
    T=system.T_ambient,
    nPorts=1,
    p=2500000);
  Modelica.Fluid.Sources.FixedBoundary sink(
    redeclare package Medium =
      Modelica.Media.Water.ConstantPropertyLiquidWater,
    p=system.p_ambient,
    T=system.T_ambient,
    nPorts=1);
  Modelica.Fluid.Valves.ValveDiscrete valve(
    redeclare package Medium =
      Modelica.Media.Water.ConstantPropertyLiquidWater,
    dp_nominal=100000,
    m_flow_nominal=10);
  Modelica.Blocks.Interfaces.BooleanInput valveInput
    "Control signal of valve state (on/off)";
  Modelica.Blocks.Interfaces.RealOutput levelOutput "Water level";
  Modelica.Blocks.Sources.RealExpression tankLevel(y=tank.level);
equation
  connect(source.ports[1],valve. port_a);
  connect(valve.port_b,tank. ports[1]);
```

```

connect(sink.ports[1], tank. ports[2]);
connect(valveInput, valve.open);
connect(tankLevel.y, levelOutput);
end ControlledWaterTankPlant;

```

A.2 Event-B Specification of the Controller

A.2.1 Context ctx

CONTEXT ctx

CONSTANTS

L

H

LT

HT

AXIOMS

axm3 : $L > 0$

axm4 : $H > L$

axm5 : $LT > L$

axm6 : $HT < H$

axm7 : $HT > LT$

axm1 : $LT = 10$

axm2 : $HT = 20$

END

A.2.2 Abstract Machine wtCtr0

MACHINE wtCtr0

SEES ctx

VARIABLES

valve

INVARIANTS

inv1 : $valve \in \text{BOOL}$

EVENTS

Initialisation

begin

act1 : $valve := \text{TRUE}$

end

```

Event openValve  $\hat{=}$ 
  any
     $l$ 
  where

  then grd1 :  $l < LT$ 

  end act1 : valve := TRUE

Event keepValve  $\hat{=}$ 
  any
     $l$ 
  where

  then grd1 :  $l \geq LT \wedge l \leq HT$ 

  end skip

Event closeValve  $\hat{=}$ 
  any
     $l$ 
  where

  then grd1 :  $l > HT$ 

  end act1 : valve := FALSE

END

```

A.2.3 Concrete Machine wtCtr1

MACHINE wtCtr1

REFINES wtCtr0

SEES ctx

VARIABLES

Read

Control

DecideClose

DecideKeep

DecideOpen

valve

level

INVARIANTS

```

typeof_Read : Read ∈ BOOL
typeof_Control : Control ∈ BOOL
typeof_DecideClose : DecideClose ∈ BOOL
typeof_DecideKeep : DecideKeep ∈ BOOL
typeof_DecideOpen : DecideOpen ∈ BOOL
distinct_states_in_sm : partition({TRUE}, {Read} ∩ {TRUE},
{Control} ∩ {TRUE}, {DecideClose} ∩ {TRUE}, {DecideKeep} ∩ {TRUE},
{DecideOpen} ∩ {TRUE})
inv1 : level ∈ L .. H
      system goal invariant

```

EVENTS

Initialisation

extended

begin

```

act1 : valve := TRUE
init_Control : Control := TRUE
init_Read : Read := FALSE
init_DecideClose : DecideClose := FALSE
init_DecideKeep : DecideKeep := FALSE
init_DecideOpen : DecideOpen := FALSE
act2 : level := 0

```

end

Event *openValve* $\hat{=}$

refines *openValve*

when

with *isin_DecideOpen* : *DecideOpen* = TRUE

then *l* : *l* = *level*

```

leave_DecideOpen : DecideOpen := FALSE
enter_Control : Control := TRUE
act1 : valve := TRUE

```

end

Event *keepValve* $\hat{=}$

refines *keepValve*

when

with *isin_DecideKeep* : *DecideKeep* = TRUE

l : *l* = *level*

```

    then
        leave_DecideKeep : DecideKeep := FALSE
        enter_Control : Control := TRUE
    end
Event closeValve ≐
refines closeValve
    when
        with isin_DecideClose : DecideClose = TRUE
        then l : l = level
        then leave_DecideClose : DecideClose := FALSE
            enter_Control : Control := TRUE
            act1 : valve := FALSE
        end
Event readLevel ≐
    any
        where l
        then grd1 : l ≥ 0
        end act1 : level := l
Event decideOpen ≐
    when
        with isin_Read : Read = TRUE
        then grd1 : level < LT
        then leave_Read : Read := FALSE
            enter_DecideOpen : DecideOpen := TRUE
        end
Event decideKeep ≐
    when
        with isin_Read : Read = TRUE
        then grd1 : level ≥ LT ∧ level ≤ HT
        then leave_Read : Read := FALSE
            enter_DecideKeep : DecideKeep := TRUE
        end
Event decideClose ≐

```



```
when
    isin_Read : Read = TRUE
    grd1 : level > HT
then
    leave_Read : Read := FALSE
    enter_DecideClose : DecideClose := TRUE
end
END
```