A Theory of Refinement of Cyber-Physical-Systems into Implementations

Bachelor's Thesis of

Daniel H. Draper

at the Department of Informatics
Institute for Theorethical Informatics

Reviewer:	Prof. Dr. Bernhard Beckert
Second reviewer:	
Advisor:	Dr. rer. nat. Mattias Ulbrich

15. April 2015 – 15. August 2015

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

Please replace with actual values

. . . . . .

(Daniel H. Draper)

Abstract

English abstract.

Zusammenfassung

Deutsche Zusammenfassung

Contents

List of Figures

List of Tables

1. Introduction

This bachelorthesis will try to formalize the following process: Replacing the abstract notion of the control program in a verified (by KeYmaera) Cyber-Physical-System (*CPS*) with an actual verified (by KeY) implementation through a form of Formal Refinement and being able to verify that the entire CPS still satisfys the required safety constraints, using both KeYmaera and KeY.

CPS are generally modelled as either Hybrid Automata or - Programs.(See ref. [**platzer2010b**]). Mostly, this means, that an abstract version of the discrete control program is modelled, as a non-deterministic assignment of one or multiple control value (See app. A.1). To replace this non-deterministic assignment with an actual implementation a certain "glue" or "coupling" has to be found to translate discrete and real continous values into each other. Glue in this case refers to a relation that encompasses both the discrete and real values. In certain cases (See chap. **??**), glue will be a concrete function, but no in general.

If we try to express the goal of this thesis in logic, we get equation 1.1. Here, ψ is the safety condition that acts both as a specification for our later implementation and the goal the program result is verified against. Glue is the aforementioned relation to be able to translate from the real into the discrete world and vice versa.

$$\begin{aligned} \textbf{If } (&\models [controlValue := *, ?\psi(controlValue) \dots]\alpha \text{ verified} \\ &\wedge \textit{"glue"(discreteVariables, continuousVariables)} \text{ verified} \\ \textbf{Then } &[controlValue := JavaProgram \dots]\alpha \text{ also verified} \end{aligned} \tag{1.1}$$

To explain this process we will take a look at the following:

I: Example-based Refinement on CPS Watertank (Example taken from KeYmaera).

II: Introduction of formalized process of using Refinement to gain a concrete implementation from a hybrid model.

III: Application of formalized process on example: CPS Gear-Backlash (See ref. [**bla**]).

1.1. Introducing Cyber-Physical-Systems

In this thesis we take a close look at **CPS**. These are systems in which a physical aspect or value is controlled by a computer (program). For example, an aircraft control system in which the computer exerts a form of speed control on the airplane would be a CPS.

In our case we take a closer look at the closely related notion of *Hybrid Systems*, in which discrete values (in the control program) and continuous values (in the physical

world) coexist. The difficulty in analyzing these kinds of systems stems from the "hybridness" of the systems: There is always some form of translation necessary to go from the program (discrete values) to the physical (continous reals) world.

two basic modelling approaches exist for hybrid systems: Hybrid Automata that are based on Non-Determnistic Finite Automata.

2. Example-based Refinement on CPS Watertank

To get a better understanding of the tasks involved in refining a hybrid model into a implementation with all necessary intermediate verification steps, we took a look at one of the out-of-the-box examples provided in the KeYmaera tutorial [**keYmaera**], a Watertank (See fig. 2.1).

The "control" part of this hybrid system is the valve, it can either drain the watertank with a rate of -2 or it can fill the tank further with a rate of 1. The goal of the entire system then is, to keep the water level y of the tank between 1 and 12. Normally this is proven by KeYmaera without actually implementing a control program. As our goal is to find a concrete implementation, we first took a look at the hybrid model of the watertank. It is provided both in the form of a hybrid automata (See fig. 2.2) and a hybrid program (See fig. 2.3).

2.1. Finding the concrete Control Value Assisgnment

In order to be able to apply Eq. 1.1 to this concrete example, the first challenge we faced was finding a spot in which a (or multiple) concrete control value is actually assigned. We call this assignment(s) the *hook* as the actual implementation will "hook" into our hybrid model at this exact point. Taking a look at the Hybrid Automata describing the Watertank (See fig. 2.2), it became obvious, that the hybrid model of the watertank required changing to be able to find our actual "hook". In the original model, a control value is never explicitly assigned, rather does the valve change its state non-deterministically, making a deterministical control program implementation impossible. Therefore, we tried to remodel the model to better serve our purpose, featuring a clear ticked hook that is called upon at deterministic times. (See fig. 2.4).

We tried to keep the model mostly equivalent to the original, preserving the 2 clock ticks of time needed for the valve to change its state from drain to fill and vice-versa. This then proved difficult, when trying to think of possible implementations, as the program hook could be called again before the valve would have actually changed its state. To simplify this problem, we decided to only model cases in which the actual program tick (so the time between two hook calls) is greater than 2 clock ticks (however long they may be). This means, that the valve or control program doesn't have to have a (possibly infinite) list of the last control values assigned, which could be the case if the program was called more frequently than the valve is able to change states.

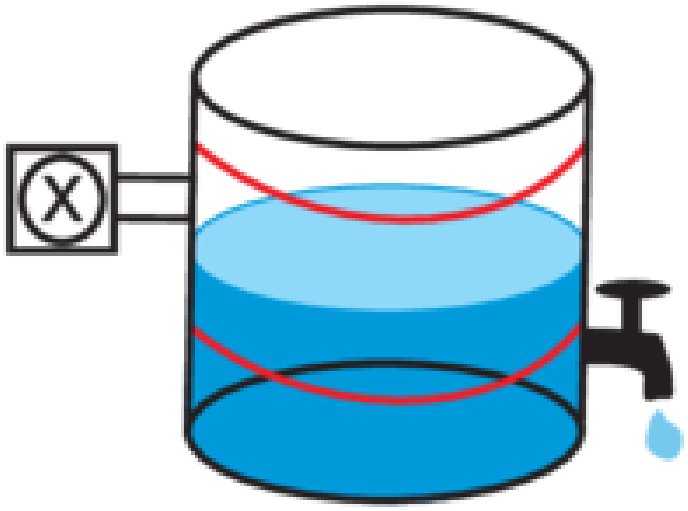

Figure 2.1.: Picture of possible Watertank configuration: Watertank that can get drained at all times and has a valve controlled by a control program [**keymaeraGuide**].

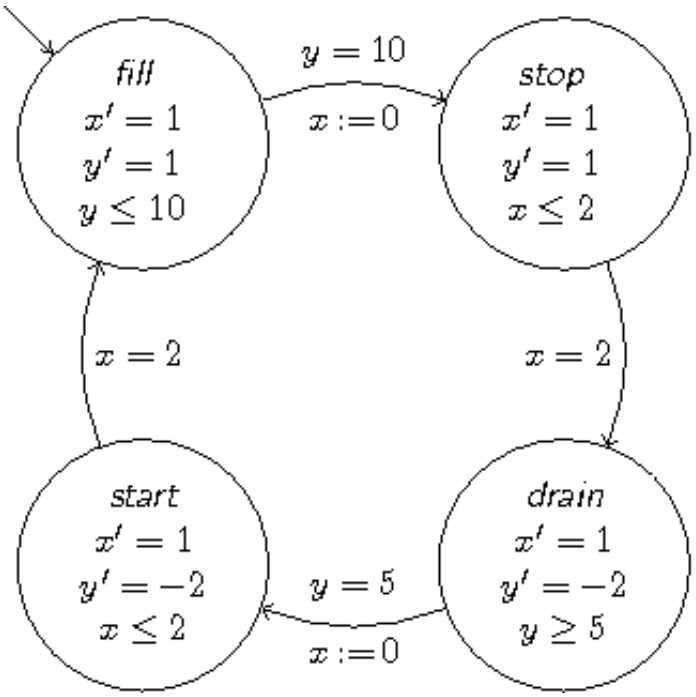

Figure 2.2.: The Watertank CPS expressed as a hybrid automata [**keymaeraGuide**].

```
/* This example does NOT use a good description style */
\programVariables {
    /* state variable declarations */
    R y, x, st;
}

\problem {
    /* initialization */
    \[  x:=0; y:=1; st:=0 \] (
     st = 0  /* initial state characterization */
   ->
     \[  /* system dynamics */
       ( /* repeat the discrete/continuous transitions */
           (?(st=0);
                 (?(y=10); x:=0; st:=1)
              ++ {x'=1,y'=1, y<=10}
           )
         ++ (?(st=1);
                 (?(x=2); st:=2)
              ++ {x'=1,y'=1, x<=2}
           )
         ++ (?(st=2);
                 (?(y=5); x:=0; st:=3)
              ++ {x'=1,y'=-2, y>=5}
            )
         ++ (?(st=3);
                 (?(x=2); st:=0)
              ++ {x'=1,y'=-2, x<=2}
           )
       )*@invariant(1<=y & y<=12
           & (st=3 -> (y>=5-2*x)) & (st=1 -> (y<=10+x)))
     \] (1<=y & y<=12)    /* safety / postcondition */
  )
 }
```

Figure 2.3.: The Watertank CPS expressed as a hybrid program [**keymaeraGuide**].

```
\functions {

    R tick;
    \external R Floor(R);
}

\programVariables {

R y; R x; R old; R new; R valve;
}

/*
invariant:
y >=1 & y <=12
*/

\problem {

 tick > 2  -> \[ x := tick; y := 1; valve := 1;

((?(x = tick ); new := *; x:= 0);
/* (? (y + 2* valve + (tick -2) * new >= 1 & y + 2* valve + (tick -2) * new <= 12 & y + 2* valve >= 1 & y + 2* valve  <= 12); */
(?(y + 2 * valve + tick * new >= 1 & y + 2 * valve + tick * new <= 12 & (new = 1 | new = -2));

((?(new != valve); {x' = 1, y' = valve & x <= 2}); if (x=2) then valve := new; {x' = 1, y' = valve & x <= tick} fi)

++ (?(!(new != valve)); {x' = 1, y' = valve & x <= tick})

))*@invariant(y >=1 & y <=12 & (valve = -2 | valve = 1) & (x = tick -> (y + 2* valve >= 1 & y + 2* valve  <= 12)))
\]
(y >= 1 & y <= 12)
}
```

Figure 2.4.: The Watertank CPS after remodelling to account for hook.

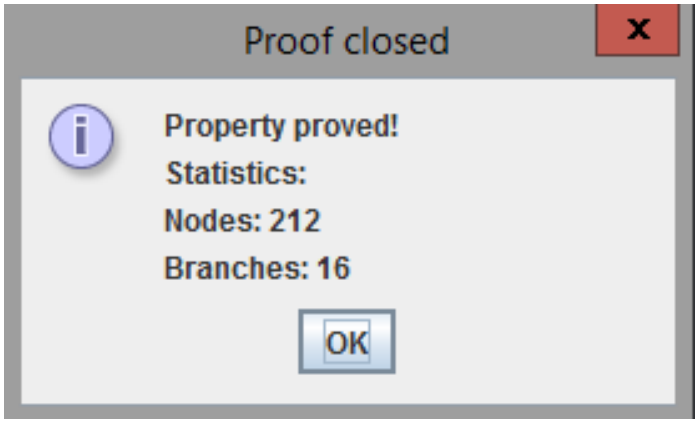

Figure 2.5.: The result of automatic verification by KeYmaera of the watertank hybrid program that includes our hook (See fig 2.4).

2.2. Refining the original Hybrid Automata

2.3. Finding the correct Program Safety Condition

The next step we attempted was finding the correct postcondition for the program hook (ψ in eq. 1.1), which would then serve as the abstraction of the java control program we would later implement. This means, that the program would be built accordingly, so that it could be verified against this postcondition. The original postcondition we devised:

$$\begin{aligned} \psi \equiv ?(&y + 2 * valve + tick - 2 * new >= 1 \ \& \\ &2 * valve * (tick - 2) * new <= 12 \ \& \\ &y + 2 * valve >= 1 \ \& \ y + 2 * valve <= 12) \end{aligned} \tag{2.1}$$

When trying to verify the entire hybrid program (see comment in line 5 of the problem in fig. 2.4), this postcondition did not work. This means, that even in such a simple CPS as this watertank control system, finding the hook postcondition was non-trivial and only with the help of KeYmaera's counterexamples did we manage to find the correct postcondition (See eq. 2.2).

$$\begin{aligned} \psi \equiv (?(&y + 2 * valve + tick * new >= 1 \ \& \\ &y + 2 * valve + tick * new <= 12 \ \& \\ &(new = 1 | new = -2)) \end{aligned} \tag{2.2}$$

KeYmaera verified our new hybrid program fully automatically (See fig. 2.5).

2.4. The (simple) Java Control Program

2.5. Finding the glue between Java and the Hybrid Model of the system

2.6. Verification based on KeYmaera

3. Introduction of formalized process of using Refinement to gain a concrete implementation from a hybrid model

What the Watertank example shows is the non-triviality of refining the hybrid model into an implementation and of the verification of all necessary parts. Overall it is obvious, that a formalized approach to the general problem presented in chapter 1 is necessary. In this chapter we present a possible formalized approach to the problem, that we deem feasible.

To aid readability we will now give an overview of the process without explanation, then detailing each step in the following sections.

1. Abstraction of the original hybrid model to better split actual control system "hook" and physical evolutions.

2. Finding the necessary safety condition of the control value for verification with KeYmaera.

3. Implementing control program according to safety condition as its specification and Verification by KeY.

4. Finding the correct "glue" between hybrid model and control program and its verification by KeYmaera.

5. Result validation: Was eq. **??** proven?

3.1. Abstracting original hybrid model to better split actual control system "hook" and physical evolutions

Most CPS we took a look at (See [**keymaera**] Tutorial, [**platzer2010b**] ...) as examples, did not have a concrete spot in which a control program could "hook" in easily. This means, that the first step in our refinement process has to be finding a suitable hook for the control program, referring to one or more non-deterministic assignments of a/multiple control values.

3.2. Finding the necessary safety condition of the control value for verification with KeYmaera

3.3. Implementing control program according to safety condition as its specification and Verification by KeY.

3.4. Finding the correct glue between hybrid model and control program and verifying it with KeYmaera

3.5. Evaluating results

4. Evaluation

…

4.1. First Section

…

4.2. Second Section

…

4.3. Third Section

…

5. Conclusion

…

A. Appendix

A.1. Images

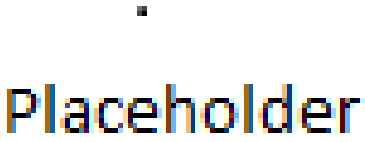

Figure A.1.: Watertank Hybrid Program and - Automata with Non-Deterministic Control Program Abstraction marked.