Formal Verification of Cyber-Physical Systems: Coping with Continuous Elements

Muhammad Usman Sanwal[1] and Osman Hasan[2]

[1] Research Center for Modeling and Simulation (RCMS)
[2] School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST),
Islamabad, Pakistan
{muhammad.usman1,osman.hasan}@seecs.nust.edu.pk

Abstract. The formal verification of cyber-physical systems is a challenging task mainly because of the involvement of various factors of continuous nature, such as the analog components or the surrounding environment. Traditional verification methods, such as model checking or automated theorem proving, usually deal with these continuous aspects by using abstracted discrete models. This fact makes cyber-physical system designs error prone, which may lead to disastrous consequences given the safety and financial critical nature of their applications. Leveraging upon the high expressiveness of higher-order logic, we propose to use higher-order-logic theorem proving to analyze continuous models of cyber-physical systems. To facilitate this process, this paper presents the formalization of the solutions of second-order homogeneous linear differential equations. To illustrate the usefulness of our foundational cyber-physical system analysis formalization, we present the formal analysis of a damped harmonic oscillator and a second-order op-amp circuit using the HOL4 theorem prover.

1 Introduction

Cyber-physical systems (CPS) [25] are characterized as computational systems, with software and digital and/or analog hardware components, that closely interact with their continuously changing physical surroundings. These days, CPS are widely being used and advocated to be used in a variety of applications ranging from ubiquitous consumer electronic devices, such as tele-operated health-care units and autonomous vehicles, to not so commonly used but safety-critical domains, such as tele-surgical robotics, space-travel and smart disaster response and evacuation. Due to the tight market windows or safety-critical nature of their applications, it has become a dire need to design error-free CPS and thus a significant amount of time is spent on ensuring the correctness of CPS designs.

Traditionally, physical and continuous aspects of a CPS are analyzed by capturing their behaviors by appropriate differential equations [31] and then solving these differential equations to obtain the required design constraints. This kind of analysis can be done using paper-and-pencil proof methods or computer based numerical techniques. Whereas, the software and digital hardware components of a CPS are usually analyzed using computer based testing or simulation methods, where the main idea is to deduce the validity of a property by observing its behavior for some test cases. However, all the above mentioned analysis techniques, i.e., paper-and-pencil proof methods, numerical methods and simulation, cannot ascertain the absence of design flaws in a design. For example, paper-and-pencil proof methods are error prone due to the human error factor. Moreover, it is quite often the case that many key assumptions of the results obtained using paper-and-pencil proof methods are in the mind of the mathematician and are not documented. Such missing assumptions may also lead to erroneous CPS designs. Similarly, computer based numerical methods cannot attain 100% accuracy as well due to the memory and computation limitations and round-off errors introduced by the usage of computer arithmetics. Thus, given the above mentioned inaccuracies, these traditional techniques should not be relied upon for the analysis of CPS, especially when they are used in safety-critical areas, such as medicine and transportation, where inaccuracies in the analysis could result in system design bugs that in turn may even lead to the loss of human life.

In the past couple of decades, formal methods [3] have been successfully used for the precise analysis of a variety of software, hardware and physical systems. The main principle behind formal analysis of a system is to construct a computer based mathematical model of the given system and formally verify, within a computer, that this model meets rigorous specifications of intended behavior. Two of the most commonly used formal verification methods are model checking [2] and higher-order-logic theorem proving [18]. Model checking is an automatic verification approach for systems that can be expressed as a finite-state machine. Higher-order-logic theorem proving, on the other hand, is an interactive approach but is more flexible in terms of tackling a variety of systems. The rigorous exercise of developing a mathematical model for the given system and analyzing this model using mathematical reasoning usually increases the chances for catching subtle but critical design errors that are often ignored by traditional techniques like paper-and-pencil based proofs, numerical methods or simulation.

Given the extensive usage of CPS in safety-critical applications, there is a dire need of using formal methods for their analysis. However, the frequent involvement of ordinary differential equations (ODEs) in their analysis is a main limiting factor in this direction. For example, ODEs are essential for modeling the motion of mechanical parts, analog circuits and control systems, which are some of the most common elements of any CPS. Thus, automatic state-based formal methods, like model checking, and automatic theorem provers cannot be used to model and analyze the true CPS models due to their inability to model continuous systems. This is the main reason why most of the formal verification work about CPS utilizes their abstracted discrete models (e.g., [28]). These limitations can be overcome by using higher-order-logic theorem proving [13] for conducting the formal analysis of CPS since the high expressiveness of higher-order logic can be leveraged upon to model elements of continuous nature. However, the main challenge in this direction is the enormous human guidance required in the formal verification of CPS due to the non-decidable nature of higher-order logic. In order to minimize this effort, we propose to develop a formal library of foundational results that can be built upon along with the automatic simplifiers to automatically reason about the correctness of CPS in a theorem prover.

In this paper, as a first step towards the proposed direction, we present the formal reasoning support for the solutions of second-order homogeneous linear differential equations [14], i.e., a simple yet the most widely used class of differential equations. In particular, we present a formal definition that can be used to specify arbitrary order homogeneous linear differential equations and the formal verification of some mathematical facts, like a couple of general solutions of second-order homogeneous linear differential equations and the quadratic formula, that allow us to reason about the correctness of their solutions in a very straightforward way. The prime advantage of these results is that they greatly minimize the user intervention for formal reasoning about differential equations and thus facilitate the usage of higher-order-logic theorem proving for verifying the solutions of differential equations for CPS. In order to demonstrate the practical effectiveness and utilization of the reported formalization, we utilize it to analyze a damped harmonic oscillator and a second-order op-amp, in this paper.

Our formalization primarily builds upon the higher-order-logic formalization of the derivative function and its associated properties. This formalization is available in a number of theorem provers like HOL4 [16], PVS [7] and CoQ [10]. Our work is based on Harrison's formalization [16] that is available in the HOL4 theorem prover [27]. The main motivations behind this choice include the past familiarity with HOL4, the availability of formalized transcendental functions, which play a key role in the reported work, and the general richness of Harrison's real analysis related theories. Though, it is important to note here that the ideas presented in this paper are not specific to the HOL4 theorem prover and can be adapted to any other LCF style higher-order-logic theorem prover as well.

The rest of the paper is organized as follows: Section 2 presents an overview of the related work. Section 3 presents a brief introduction to theorem proving and the HOL4 theorem prover. This is followed by a brief introduction to the HOL4 formalization of the derivative and integration functions in Section 4. In Section 5, we present the formalization of the solutions of the second-order homogeneous linear differential equation. The illustrative examples are presented in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Formal methods have been extensively used these days for analyzing CPS due to their ever increasing usage in various safety and financial-critical domains. Zhang et. al [21] proposed to use formal specification for CPS in order to reduce the infinite set of test parameters in a finite set. Similarly, the aspect-oriented programming based on the UML and formal methods is utilized for QoS modeling of CPS in [20]. Moreover, in order to formally specify CPS along with their continuous aspects, a combination of formal methods Timed-CSP, ZimOO and differential (algebraic) equations is used in [30]. Even though such rigorous formal specifications allow us to catch bugs in the early stages of the design but they do not guarantee error-free analysis since the analysis or verification is not based on formal methods.

For formal verification of CPS, model-checking has been frequently explored. For example, Akella [1] proposed to dicretize the events causing the change of flow and thus model the CPS as a deterministic state model with discrete values of flow within its physical components. This model is then used to formally verify insecure interactions between all possible behaviors of the given CPS using model checking. Similarly, Bu et. al [6] used hybrid model checking for verifying CPS. However, this verification is also not based on true continuous models of the system and instead a short-run behavior of the model is observed by providing numerical values of various parameters in order to reduce the state-space. A statistical model checker has been recently utilized to analyze some aspects of CPS [9]. However, this approach also suffers from the classical model checking issues, like the state-space explosion and inability to reason about generic mathematical relations. Thus the model checking approach, even though is capable of providing exact solutions, is quite limited in terms of handling true continuous models of CPS and thus various abstractions [28] have to be used for attaining meaningful results. The accuracy of the analysis is thus compromised, which is undesirable in the case of analyzing safety-critical applications of CPS.

Higher-order-logic theorem proving is capable of overcoming all the above mentioned problems. Atif et. al [22] used the HOL4 theorem prover for the probabilistic analysis of cyber-physical transportation systems. However, their focus was only on the formal verification of probabilistic aspects of CPS and they did not tackle the continuous aspects, especially the ones that require to be modeled by ODEs, which is the main focus of the current paper.

3 Preliminaries

In this section, we give a brief introduction to theorem proving in general and the HOL theorem prover in particular. The intent is to introduce the main ideas behind this technique to facilitate the understanding of this paper for the CPS research community.

3.1 Theorem Proving

Theorem proving [13] is a widely used formal verification technique. The system that needs to be analyzed is mathematically modeled in an appropriate logic and the properties of interest are verified using computer based formal tools. The use of formal logics as a modeling medium makes theorem proving a very flexible verification technique as it is possible to formally verify any system that can be described mathematically. The core of theorem provers usually consists of some well-known axioms and primitive inference rules. Soundness is assured as every new theorem must be created from these basic or already proved axioms and primitive inference rules.

The verification effort of a theorem in a theorem prover varies from trivial to complex depending on the underlying logic [15]. For instance, first-order logic [12] is restricted to propositional calculus and terms (constants, function names and free variables) and is semi-decidable. A number of sound and complete first-order logic automated reasoners are available that enable completely automated proofs. More expressive logics, such as higher-order logic [5], can be used to model a wider range of problems than first-order logic, but theorem proving for these logics cannot be fully automated and thus involves user interaction to guide the proof tools. For analyzing continuous aspects of systems, we have to use higher-order logic as first-order logic is not expressive enough to represent real numbers and calculus fundamentals.

3.2 HOL Theorem Prover

HOL is an interactive theorem prover developed by Mike Gordon at the University of Cambridge for conducting proofs in higher-order logic. It utilizes the simple type theory of Church [8] along with Hindley-Milner polymorphism [23] to implement higher-order logic. HOL has been successfully used as a verification framework for both software and hardware as well as a platform for the formalization of pure mathematics.

Secure Theorem Proving In order to ensure secure theorem proving, the logic in the HOL system is represented in the strongly-typed functional programming language ML [24]. An ML abstract data type is used to represent higher-order logic theorems and the only way to interact with the theorem prover is by executing ML procedures that operate on values of these data types. The HOL core consists of only 5 basic axioms and 8 primitive inference rules, which are implemented as ML functions. Soundness is assured as every new theorem must be verified by applying these basic axioms and primitive inference rules or any other previously verified theorems/inference rules.

Terms There are four types of HOL terms: constants, variables, function applications, and lambda-terms (denoted function abstractions). Polymorphism, types containing type variables, is a special feature of higher-order logic and is thus supported by HOL. Semantically, types denote sets and terms denote members of these sets. Formulas, sequences, axioms, and theorems are represented by using terms of Boolean types.

Theories A HOL theory is a collection of valid HOL types, constants, axioms and theorems, and is usually stored as a file in computers. Users can reload a HOL theory in the HOL system and utilize the corresponding definitions and theorems right away. The concept of HOL theory allows us to build upon existing results in an efficient way without going through the tedious process of regenerating these results using the basic axioms and primitive inference rules.

HOL theories are organized in a hierarchical fashion. Any theory may inherit types, definitions and theorems from other available HOL theories. The HOL system prevents loops in this hierarchy and no theory is allowed to be an ancestor and descendant of a same theory. Various mathematical concepts have been formalized and saved as HOL theories by the HOL users. These theories are available to a user when he first starts a HOL session. We utilized the HOL theories of Booleans, lists, sets, positive integers, *real* numbers and calculus in our work. In fact, one of the primary motivations of selecting the HOL theorem prover for our work was to benefit from these built-in mathematical theories.

Writing Proofs HOL supports two types of interactive proof methods: forward and backward. In forward proof, the user starts with previously proved theorems and applies inference rules to reach the desired theorem. In most cases, the forward proof method is not the easiest solution as it requires the exact details of a proof in advance. A backward or a goal directed proof method is the reverse of the forward proof method. It is based on the concept of a *tactic*; which is an ML function that breaks goals into simple sub-goals. In the backward proof method, the user starts with the desired theorem or the main goal and specifies tactics to reduce it to simpler intermediate sub-goals. Some of these intermediate sub-goals can be discharged by matching axioms or assumptions or by applying built-in decision procedures. The above steps are repeated for the remaining intermediate goals until we are left with no further sub-goals and this concludes the proof for the desired theorem.

The HOL theorem prover includes many proof assistants and automatic proof procedures [15] to assist the user in directing the proof. The user interacts with a proof editor and provides it with the necessary tactics to prove goals while some of the proof steps are solved automatically by the automatic proof procedures.

4 Derivatives and Integrals in HOL4

Harrison [16] formalized the *real number theory* along with the fundamentals of calculus, such as limits of a function, derivatives and integrals and verified most of their classical properties in HOL4. The derivative of a function f, of data type (`real` $\rightarrow$ `real`), is defined as follows [16]:

Definition 1: *Derivative of a Function (Relational Form)*
$\vdash$ ∀ `f l x. (f diffl l) x = ((`λ `h.(f (x + h) - f x) / h)` → `l) (0)`

where (`f` → `y0`)(`x0`) represents the HOL4 definition of limit of a function f $lim_{(x \rightarrow x0)} f(x) = y0$. Definition 1 provides the derivative of a function f at point x as the limit value of $\frac{f(x+h)-f(x)}{h}$ when h approaches 0, which is the standard mathematical definition of the derivative function. Now, the differentiability of a function f is defined as the existence of its derivative [16].

Definition 2: *Differentiability of a Function*

```
⊢ ∀ f x. f differentiable x = ∃l. (f diffl l) (x)
```

A functional form of the derivative, which can be used as a binder, is also defined using the Hilbert choice operator `@` as follows [16]:

Definition 3: *Derivative of a Function (Functional Form)*

```
⊢ ∀ f x. deriv f x = @l. (f diffl l) x
```

The function `deriv` accepts two parameters f and x and returns the derivative of function f at point x.

The above mentioned definitions associated with the derivative function have been accompanied by the formal verification of most of their classical properties, such as uniqueness, linearity and composition [16]. Moreover, the derivatives of some commonly used transcendental functions have also been verified. For example, the derivative of the Exponential function has been verified as follows:

Theorem 1: *Differential of the Exponential Function*

```
⊢ ∀ g m x. ((g diffl m) x ⇒
    ((λ.x. exp (g x)) diffl (exp (g x) * m)) x)
```

where `exp x` represents the exponential function e^x and (λx.f(x)) represents the lambda abstraction function which accepts a variable x and returns $f(x)$.

Similarly, the Gauge integral has also been formalized as a function `Dint (a,b) f k` [16], which mathematically describes $\int_a^b \mathtt{f(x)\ dx} = \mathtt{k}$. The corresponding functional form is given as follows:

Definition 4: *Integral of a Function (Functional Form)*

```
⊢ ∀ a b f . integral (a,b) f = @k. Dint (a,b) f k
```

Many interesting properties of integration have been formally verified in [16]; one of them being the second fundamental theorem of calculus.

Theorem 2: *Second Fundamental Theorem of Calculus*

```
⊢ ∀ f f' a b. (a ≤ b) ∧
  (∀x. a ≤ x ∧ x ≤ b ⇒ (f diffl f'(x))(x))
    ⇒ Dint(a,b) f' (f(b) - f(a))
```

We build upon the above mentioned formalization to develop formal reasoning support for second-order homogeneous linear differential equations in the next section.

5 Second-order Homogeneous Differential Equations

A second-order homogeneous linear differential equation can be mathematically expressed as follows:

$$p_2(x)\frac{d^2y(x)}{dx} + p_1(x)\frac{dy(x)}{dx} + p_0(x)y(x) = 0 \tag{1}$$

where terms p_i represent the coefficients of the differential equation defined over a function y. The equation is linear because (i) the function y and its derivatives appear only in their first power and (ii) the products of y with its derivatives are also not present in the equation. By finding the solution of the above equation, we mean to find functions that can be used to replace the function y in Equation (1) and satisfy it.

We proceed to formally represent Equation (1) by first formalizing an n$^{\mathtt{th}}$-order derivative function as follows:

Definition 5: *N$^{\mathtt{th}}$-order Derivative of a Function*
```
⊢ (∀ f x. n_order_deriv 0 f x = f x) ∧
  (∀f x n.n_order_deriv (n+1) f x = n_order_deriv n (deriv f x) x)
```

The function `n_order_deriv` accepts an integer n that represents the order of the derivative, the function f that represents the function that needs to be differentiated, and the variable x that is the variable with respect to which we want to differentiate the function f. It returns the n$^{\mathtt{th}}$-order derivative of f with respect to x. Now, based on this definition, we can formalize the left-hand-side (LHS) of an n$^{\mathtt{th}}$-order differential equation in HOL4 as the following definition.

Definition 6: *LHS of a N$^{\mathtt{th}}$-order Differential Equation*
```
⊢ ∀ P y x. diff_eq_lhs P y n x =
    sum(0,n)(λn.(EL n L x) * (n_order_deriv n y x))
```

The function `diff_eq_lhs` accepts a list P of coefficient functions corresponding to the p_i's of Equation (1), the differentiable function y, the order of differentiation n and the differentiation variable x. It utilizes the HOL4 functions `sum (0,n) f` and `EL n L`, which correspond to the summation $(\sum_{i=0}^{n-1} f_i)$ and the `n`$^{\mathtt{th}}$ element of a list L_n, respectively. It generates the LHS of a differential equation of n$^{\mathtt{th}}$ order with coefficient list P. The second-order differential equation of Equation (1) can now be formally modeled by instantiating variable n of Definition 6 by number 3.

If the coefficients p_i's of Equation (1) are constants in terms of the differentiation variable x then, using the fact that the derivative of the exponential function $y = e^{rx}$ (with a constant r) is a constant multiple of itself $dy/dx = re^{rx}$, we can obtain the following solution of Equation (1):

$$Y(x) = c_1e^{r_1x} + c_2e^{r_2x} \tag{2}$$

where c_1 and c_2 are arbitrary constants and r_1 and r_2 are the roots of the auxiliary equation $p_2r^2 + p_1r^1 + p_0 = 0$ [31]. In this paper, we formally verify this result which plays a key role in formal reasoning about the solutions of second-order homogeneous linear differential equations in a higher-order-logic theorem prover.

Theorem 3: *General Solution of a Homogeneous Linear Differential Equation*

```
⊢ ∀ a b c c1 c2 r1 r2 x.
  (c + (b * r1) + (a * (r1 pow 2)) = 0) ∧
  (c + (b * r2) + (a * (r2 pow 2)) = 0) ⇒
  (diff_eq_lhs (const_list [c; b; a])
    (λx. c1 * (exp (r1 * x) + c2 * (exp (r2 * x)) 3 x = 0)
```

where $[c; b; a]$ represents the list of constants corresponding to the coefficients p_0, p_1 and p_2 of Equation (1), $r1$ and $r2$ represent the roots of the corresponding auxiliary equation as given in the assumptions, $c1$ and $c2$ are the arbitrary constants and x is the variable of differentiation. The function `const_fn_list` used in the above theorem transforms a constant list to the corresponding constant function list recursively as follows:

Definition 7: *Constant Function List*

```
⊢ (const_fn_list [] = []) ∧
(∀ h t. const_fn_list (h::t) = (λ(x:real). h) :: (const_fn_list t))
```

The function `diff_eq_lhs` permits coefficients that are functions of the variable of differentiation but Theorem 3 is valid only for constant coefficients. Thus, using `const_fn_list` we provide the required type for the coefficient list of the function `diff_eq` while fulfilling the requirement of Theorem 3. The formal reasoning about Theorem 3 is primarily based on Theorem 1 and the linearity property of higher-order derivatives, which has been verified in our work for *class* C^n functions, i.e., the functions for which the first n derivatives exist for all x as the following higher-order-logic theorem:

Theorem 4: *Linearity of* n^{th}*-order Derivative*

```
⊢ ∀ f g x a b.
  (∀m x. m ≤ n ⇒ (λx. n_order_deriv m f x) differentiable x) ∧
  (∀m x. m ≤ n ⇒ (λx. n_order_deriv m g x) differentiable x) ⇒
    (n_order_deriv n (λx. a * f x + b * g x) x =
      a * n_order_deriv n f x + b * n_order_deriv n g x)
```

where variables a and b represent constants with respect to variable x. The formal reasoning about Theorem 4 involves induction on variable n, which represents the order of differentiation, and is primarily based on the linearity property of the first order derivative function [16].

If the roots of Equation (1) are real and repeated then we can obtain the following solution:

$$Y(x) = c_1 e^{rx} + c_2 x e^{rx} \tag{3}$$

where c_1 and c_2 are arbitrary constants and r is the real and repeated root of the auxiliary equation $p_2 r^2 + p_1 r^1 + p_0 = 0$ [31]. Just like Theorem 3, we also formally verified that this solution satisfies Equation (1).

Theorem 5: *Differential Equation with repeated roots*

```
⊢ ∀ a b c c1 c2 r1 r2 x.
  (c + (b * r) + (a * r²) = 0) ∧ (b² - (4 * a * c) = 0) ∧ b ≠ 0 ⇒
      (diff_eq_lhs (const_list [c; b; a])
        (λy. c1 * (exp (r * y)) + c2 * (y * exp (r * y))) 3 x = 0)
```

where r represents the sole root of the corresponding auxiliary equation and the rest of the variables are the same as Theorem 3. The formal reasoning about Theorem 5 is also mainly based on Theorems 1 and 4 and the well-known quadratic formula which was also formally verified in our development as the following theorem in our development.

Theorem 6: *Quadratic Formula*

```
⊢ ∀ a b c x. (a ≠ 0) ∧ (4 * a * c < b pow 2) ⇒
aux_eq_roots_list [((-b + sqrt (b pow 2 - 4 * a * c)) / (2 * a));
    ((-b - sqrt (b pow 2 - 4 * a * c)) / (2 * a))]
      (const_fn_list [a; b; c]) x
```

where the functions `sqrt` and `pow` represent the square-root and square of a real number, respectively. The theorem essentially says that the roots of the auxiliary equation ax^2+bx+c are given by the first list argument of the function `aux_eq_roots_list`. The assumption `(4 * a * c < b pow 2)` guarantees that the roots are always real.

We will build upon the results presented in this section to formally reason about the applications in Section 6 of this paper.

6 Applications

In this section, we formally analyze two widely used continuous components of CPS, i.e., a damped harmonic oscillator and a second-order op-amp circuit. The first one being a widely used mechanical model for CPS whereas the later is a widely used analog component of CPS.

6.1 Damped Harmonic Oscillator

Behaviors of many CPS are mathematically equivalent to harmonic oscillators, that is, these systems can be described by an ODE that is similar to the ODE of the damped harmonic oscillator. Some prominent examples include the spring mass system and the classical RLC circuit [11]. Therefore, the analysis of the damped harmonic oscillator, depicted in Figure 1, has gained interest from the CPS community [4,26]. The ODE of a damped harmonic oscillator can be expressed as

$$\frac{d^2x}{dt^2} + z\frac{dx}{dt} + w^2x = 0 \tag{4}$$

Where x is the extension produced, $w = \sqrt{(k/m)}$ represents the angular frequency of the oscillator in terms of spring constant k and the mass of spring m

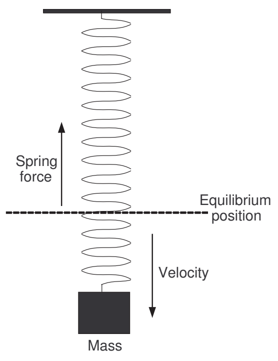

Fig. 1. Damped Harmonic Oscillator

and $z = \sqrt{(b/m)}$ represents the damping ratio in terms of the damping coefficient b and the mass of spring m.

Based on the formalization of the last section, the solution of the above differential equation can be formally verified as the following theorem:

Theorem 7: *Damped Harmonic Oscillator*

```
⊢ ∀ w z c1 c2 x. (z^2 = w^2) ⇒
(diff_eq_lhs (const_list [w^2; z; 1])
  (λx.c1 * exp((-z + sqrt (z^2 - w^2)) * x) +
      c2 * exp((-z - sqrt (z^2 - w^2)) * x)) x = 0)
```

This theorem is formally verified primarily using Theorem 3. The assumptions of Theorem 7 declare the relationships between the various parameters that are required for the solution to hold. This is one of strengths of the proposed theorem proving based verification as all the assumptions have to be explicitly stated besides the theorem for its formal verification. Thus, there is no chance of missing a critical assumption which often occurs in paper-and-pencil proof methods. It is also important to note the generic and continuous nature of Theorem 7 as all the variables are of type `real` and they are universally quantified. Such results cannot be obtained via state-based formal methods tools.

6.2 Second-order Op-Amp

As a second example, consider a second-order op-amp circuit (Figure 2), which has a wide range of applications in oscillators, filters, audio buffers and line drivers [19,29]. Thus, it is a widely used continuous component of many CPS.

The behavior of this circuit can be modeled as the following differential equation [19]:

$$\frac{d^2v}{dt^2} - (\frac{1}{R^2C^2})v = 0 \tag{5}$$

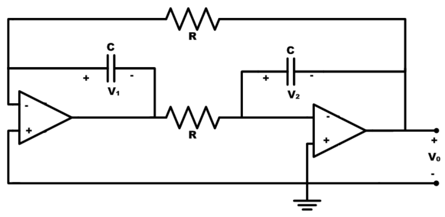

Fig. 2. Second-order Op-Amp Circuit

We formally verified its solution, i.e., an expression for the voltage v, as the following theorem.

Theorem 8: *Second-order Op-Amp Circuit*
$\vdash$ $\forall$ `R L c1 c2. (R * C` $\neq$ `0)` $\Rightarrow$
`(diff_eq_lhs (const_list [-1/(`R^2 `*` C^2`);0;1])`
`(`λ`x. c1 * exp ((1/R*C) * x) + c2 * (x * exp ((1/R*C) * x)) x = 0))`

The proof of the above theorem is based on Theorem 5, which presents the formally verified solution of the homogeneous linear differential equation with real and repeated roots, along with some arithmetic reasoning, which can be done in an automatic manner using the HOL4 arithmetic simplifiers.

The proof script for all the theorems, presented in this section, is composed of just 300 lines approximately. This is far less than the proof script for the formalization, presented in the previous section, which is more than 3500 lines of HOL code. This fact clearly indicates the usefulness of our foundational formalization presented in Sections 5 of this paper. Just like the case studies, presented in this section, our formalization results can be utilized to automatically verify interesting properties of a wide variety of CPS in a straight-forward manner and the results would be guaranteed to be correct due to the inherent soundness of theorem proving.

7 Conclusions

In this paper, we propose to use higher-order-logic theorem proving to analyze continuous aspects of CPS. Due to the high expressiveness of the underlying logic, we can formally model the continuous components of CPS while capturing their true behavior and the soundness of theorem proving guarantees correctness of results. To the best of our knowledge, these features are not shared by any other existing CPS analysis technique. The main challenge in the proposed approach is the enormous amount of user intervention required due to the undecidable nature of the logic. We propose to overcome this limitation by formalizing the foundational mathematical theories so that these available results can be built upon to minimize user interaction. As a first step towards this direction, we presented the formalization of the solutions of second-order homogenous linear differential equations in this paper. Based on this work, we are able to formally analyze the damped harmonic oscillator and the second order op-amp circuit, which are both quite frequently encountered in CPS analysis, in a very straightforward way.

The proposed approach opens the doors to many new directions of research. We are working on developing reasoning support for non-homogeneous linear differential equations. Moreover, the calculus theories available in HOL-Light [17] are based on multivariate real numbers and thus can model complex numbers. Our formalization can be ported in a very straight-forward manner to this formalization of complex numbers in HOL-Light, which would enable handling the formal analysis of CPS that can be modeled in the complex plane only.

Acknowledgment

This work was supported by the National Research Program for Universities grant (number 1543) of Higher Education Commission (HEC), Pakistan.

References

1. R. Akella and B.M. McMillin. Model-Checking BNDC Properties in Cyber-Physical Systems. In *Computer Software and Applications Conference*, pages 660 –663, 2009.
2. C. Baier and J. Katoen. *Principles of Model Checking.* MIT Press, 2008.
3. P.P. Boca, J.P. Bowen, and J.I. Siddiqi. *Formal Methods: State of the Art and New Directions.* Springer, 2009.
4. D. Broman, E. A. Lee, S. Tripakis, and M. Toerngren. Viewpoints, Formalisms, Languages, and Tools for Cyber-Physical Systems. In *6th International Workshop on Multi-Paradigm Modeling*, 2012.
5. C.E. Brown. *Automated Reasoning in Higher-order Logic.* College Publications, 2007.
6. L. Bu, Q. Wang, X. Chen, L. Wang, T. Zhang, J. Zhao, and X. Li. Towards Online Hybrid Systems Model Checking of Cyber-Physical Systems' Time-Bounded Short-Run Behavior. *SIGBED*, (2):7–10, 2011.
7. R. W. Butler. Formalization of the Integral Calculus in the PVS Theorem Prover. *Journal of Formalized Reasoning*, 2(1):1–26, 2009.
8. A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.
9. E. M. Clarke and P. Zuliani. Statistical model checking for cyber-physical systems. In *Automated Technology for Verification and Analysis*, volume 6996 of *LNCS*, pages 1–12. Springer, 2011.
10. L. Cruz-Filipe. *Constructive Real Analysis: a Type-Theoretical Formalization and Applications.* PhD thesis, University of Nijmegen, April 2004.
11. Y. Daneshbod and J. Latulippe. A Look at Damped Harmonic Oscillators through the Phase Plane. *Teaching Mathematics and its Applications*, 30(2).
12. M. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer, 1996.

13. M.J.C. Gordon. Mechanizing Programming Logics in Higher-0rder Logic. In *Current Trends in Hardware Verification and Automated Theorem Proving*, pages 387–439. Springer, 1989.
14. G.Strang. *Calculus.* Wellesley College, second edition, 2009.
15. J. Harrison. Formalized Mathematics. Technical Report 36, Turku Centre for Computer Science, 1996.
16. J. Harrison. *Theorem Proving with the Real Numbers.* Springer-Verlag, 1998.
17. J. Harrison. A HOL theory of Euclidean space. In *Theorem Proving in Higher Order Logics*, volume 3603 of *LNCS*, pages 114–129. Springer, 2005.
18. J. Harrison. *Handbook of Practical Logic and Automated Reasoning.* Cambridge University Press, 2009.
19. K.Alexander and M.N. O. Sadiku. *Fundamentals of Electric Circuits.* McGraw-Hill, 2008.
20. J. Liu and L. Zhang. QoS Modeling for Cyber-Physical Systems using Aspect-Oriented Approach. In *Networking and Distributed Computing (ICNDC), 2011 Second International Conference on*, pages 154 –158, 2011.
21. L.Zhang, J. Hu, and W . Yu. Generating Test Cases for Cyber Physical Systems from Formal Specification. pages 97–103. Springerl, 2011.
22. A. Mashkoor and O. Hasan. Formal Probabilistic Analysis of Cyber-Physical Transportation Systems. In *ICCSA (3)*, pages 419–434, 2012.
23. R. Milner. A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, 17:348–375, 1977.
24. L.C. Paulson. *ML for the Working Programmer.* Cambridge University Press, 1996.
25. R. Rajkumar, I. Lee, L. Sha, and J. J. Stankovic. Cyber-Physical Systems: The next Computing Revolution. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 731 –736, 2010.
26. J. Shi, J. Wan, H. Yan, and H. Suo. A survey of Cyber-Physical Systems. In *Wireless Communications and Signal Processing (WCSP)*, pages 1 –6, 2011.
27. K. Slind and M. Norrish. A Brief Overview of HOL4. In *Theorem Proving in Higher-order Logics*, volume 5170 of *LNCS*, pages 28–32. Springer, 2008.
28. R. A. Thacker, K. R. Jones, C. J. Myers, and H. Zheng. Automatic Abstraction for Verification of Cyber-Physical Systems. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 12–21. ACM, 2010.
29. W.Jung. *Op Amp Applications Handbook.* Newnes, 2004.
30. L. Zhang. Aspect Oriented Formal Techniques for Cyber Physical Systems. *journal of software*, 7(4):823–834, 2012.
31. D.G. Zill, W.S. Wright, and M.R. Cullen. *Advanced Engineering Mathematics.* Jones and Bartlett Learning, fourth edition, 2009.