Formal Analysis of Steady State Errors in Feedback Control Systems using HOL-Light

Osman Hasan
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
H-12, Islamabad 44000, Pakistan
Email: osman.hasan@seecs.nust.edu.pk

Muhammad Ahmad
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
H-12, Islamabad 44000, Pakistan
Email: muhammad.ahmad@seecs.nust.edu.pk

***Abstract*—The accuracy of control systems analysis is of paramount importance as even minor design flaws can lead to disastrous consequences in this domain. This paper provides a higher-order-logic theorem proving based framework for the formal analysis of steady state errors in feedback control systems. In particular, we present the formalization of control system foundations, like transfer functions, summing junctions, feedback loops and pickoff points, and steady state error models for the step, ramp and parabola cases. These foundations can be built upon to formally specify a wide range of feedback control systems in higher-order logic and reason about their steady state errors within the sound core of a theorem prover. The proposed formalization is based on the complex number theory of the HOL-Light theorem prover. For illustration purposes, we present the steady state error analysis of a solar tracking control system.**

I. Introduction

All dynamic systems, ranging from an alarm clock to satellites, require a control system to guide them in performing the desired tasks in a variety of harsh environments. Control system design is a well-established field of engineering and the main goal in this domain is to design a dedicated control system for a given system such that the two integrated together exhibit the desired behavior and satisfy strict constraints on allowable error margins and stability.

Traditionally, control system design has been done based on mathematical analysis using paper-and-pencil proof methods. Mathematical models of the given system and the corresponding control system are constructed using differential equations, in the time or Laplace domain. The advantage of the later is that it facilitates modeling the main system in terms of transfer functions of its sub-systems, as a block diagram. This block diagram representation allows us to obtain the net transfer function of the system by manipulating the transfer functions of the subsystems using a set of predefined rules [14]. The net transfer function, of the entire system, is then used to judge various characteristics of the system, such as stability and steady-state errors. Various graphical representations, such as Root Locus technique, Bode plots and Nichole charts are also used after the net transfer function of the system is obtained to judge various design parameters. However, considering the complexity of present age control systems, such kind of manual analysis is notoriously difficult, if not impossible, and is 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. These missing assumptions may also lead to erroneous designs.

978-3-9815370-0-0/DATE13/2013 EDAA

With the advent of computers, many computer-aided design tools based on the principles of numerical methods and simulation have been introduced for modeling and analysis of control systems. For example, MathWorks Simulink [13] provides a graphical interface to model a control system as the standard engineers' block-diagram and then analyze it. Due to the reliable and efficient bookkeeping characteristic of computers, large control systems can be analyzed using these methods. However, these methods cannot attain 100% accuracy due to the memory and computation limitations and the usage of computer arithmetics, like floating or fixed point numbers. The effect of these small round-off errors per computation becomes very significant while analyzing control systems, which usually involve iterations in the range of billions. Another alternative for analyzing control systems is computer algebra systems such as Mathematica [11]. These methods are very efficient for computing mathematical solutions symbolically, but they are not reliable as well [6] due to the presence of unverified huge symbolic manipulation algorithms in their core, which are quite likely to contain bugs. Thus, given the above mentioned inaccuracies, these traditional techniques should not be relied upon for analyzing control systems used in safety-critical applications, such as surgical robotics or automatic transportation systems, where inaccuracies in the analysis could result in system design bugs that may even lead to the loss of human life.

Formal methods are capable of overcoming the above mentioned limitations and there is growing trend toward using them for analyzing control systems [16], [2]. Arthan developed a tool, called ClawZ [3], that allows us to translate discrete-time system models from MathWorks Simulink into Z language specifications, which are in turn verified to be equivalent to the controller implementation using Ada language in Proof-Prover. Mahony followed a similar approach by providing the support of modeling and analyzing feedback control systems using the DOVE environment [12]. Tiwari used model checking to analyze dynamic systems by abstracting the behavior of

the system to a state-space model [17]. Herencia-Zapana [9] proposed to formally analyze control software properties by first expressing stability proofs as C code annotations and then translating them to PVS proof obligations and automatically verifying them. However all these pioneering frameworks are based on automatic formal verification tools and thus cannot be used to analyze the exact models of real-world control systems that are always continuous in nature.

In order to formally model and analyze continuous models of control systems, Boulton et al. provided some reasoning support for verifying frequency response of continuous-time control systems using Hoare logic in HOL98 theorem prover [5]. The main idea is to reason about the gain and phase relationships of a control system using the gain and phase relationships of its subsystems in the block diagram. This framework does not provide generic functions to model arbitrary block diagrams for control systems and and also lacks reasoning support for complex number analysis principles, such as limits and summation, which are essential to reason about many control system design related parameters, such as steady state errors and stability. In order to overcome these shortcomings, Boulton et al [4] proposed to use automated symbolic methods to replace the classical graphical charts, such as Nichole and Bode plots along with their formal models. Based on this principle, the authors developed a prototype tool using Maple and the PVS system. Maple is used to compute the verification conditions for the given control system and PVS is used to discharge these conditions using theorem proving principles. Due to the usage of computer algebra system Maple the accuracy of the analysis is again somewhat compromised as has been mentioned above.

This paper is primarily oriented towards overcoming the above mentioned limitations. Our main goal is to develop a generic higher-order-logic theorem proving framework for conducting formal analysis of continuous models of control systems. Since the discrete domain is a subset of the continuous one, the proposed development would be equally good for analyzing discrete systems as well. We propose to build upon the higher-order-logic formalization of complex number theory [8] that is available in the Hol-Light theorem prover. Rather than specifying a new data type for complex numbers, they are modeled as a two dimensional vector, with data type $\mathtt{R}^2$, in the Euclidean space [7]. The main advantage of this choice is that all the topological and analytic formalization developed for vectors is inherited by the complex numbers. These rigorous foundations greatly facilitate the objective of our work. As a first step towards our main goal, we provide the formalization of a set of definitions that allow us to formally express any block diagram, corresponding to a single-input-single-output control system in the Laplace-domain. Besides that, we also provide some key formally verified theorems that facilitate the formal reasoning about simplifying the block diagram of a complete control systems to a single transfer function. Moreover, we also provide formal definitions and formal reasoning support for conducting the formal steady state error analysis of control systems, which is primarily the limiting case of the difference between the required output and a predefined test input. To the best of our knowledge, this is the first time that formal reasoning support for these foundations has been reported in the open literature. The prime advantage of these results is that they greatly minimize the user intervention for formal reasoning about the properties of control systems in a higher-order-logic theorem prover. In order to demonstrate the practical effectiveness and utilization of the reported formalization, we utilize it to analyze a solar tracking control system [10].

II. Block Diagrams of Control Systems

This section provides a set of formal definitions corresponding to the basic building blocks of control systems given in Fig. 1. The main advantage of these definitions is that they can be used to formalize block diagram of any control system.

In the Laplace domain, the transfer function of two subsystems connected in *series* is equivalent to the product of their individual transfer functions, as shown in Fig. 1.a. This can be formalized for arbitrary serially connected subsystems as:

Definition 1: $\forall\ \mathtt{X}_i.\ \mathtt{series}\ [\mathtt{X}_1;\ \mathtt{X}_2;..;\ \mathtt{X}_N] = \prod_{i=1}^{N} X_i$

The function `series` accepts a list of complex numbers, which represent the transfer functions of individual subsystems, and recursively returns their product.

The *summation junction* is an addition module that adds the transfer functions of all the incoming branches, as shown in Fig. 1.b. We formalized it for a list of complex numbers, representing the transfer functions of each branch, as:

Definition 2: $\forall\ \mathtt{X}_i.\mathtt{sum\_j}\ [\mathtt{X}_1;\ \mathtt{X}_2;..;\ \mathtt{X}_N] = \sum_{i=1}^{N} X_i$

The *pickoff point* represents a submodule connected to a parallel branch of submodules (Fig. 1.c). The function `pickoff`, given below, accepts a complex number `a`, corresponding to the transfer function of the first module, and a list of complex numbers, corresponding to the transfer functions of the submodules in the branches, and returns a list of complex numbers corresponding to the equivalent block diagram.

Definition 3: $\forall\ \mathtt{A}\ \mathtt{X}_i.\ \mathtt{pickoff\ A}\ [\mathtt{X}_1;\ \mathtt{X}_2;...;\ \mathtt{X}_N] = [\mathtt{A} * \mathtt{X}_1;\ \mathtt{A} * \mathtt{X}_2;\ \ldots\ ;\ \mathtt{A} * \mathtt{X}_N]$

The *feedback* block, shown in Fig. 1.d, is the foremost element required to model closed loop control systems. Due to the feedback signal, it basically represents an infinite summation of a number of branches that are composed of serially connected submodules, as shown in Fig. 1.d. We first formalize the net transfer function of each branch as follows:

Definition 4: $\forall\ \mathtt{a\ b\ N.branch\ a\ b\ N} = \prod_{i=0}^{N} \mathtt{series[a;b]}$

The function `branch` accepts the forward path transfer function `a`, the feedback path transfer function `b` and the sequence number of the branch `n`. It returns the net transfer function of the n^{th} branch as a single complex number. Now, we can model the behavior of the feedback loop in HOL-Light as follows:

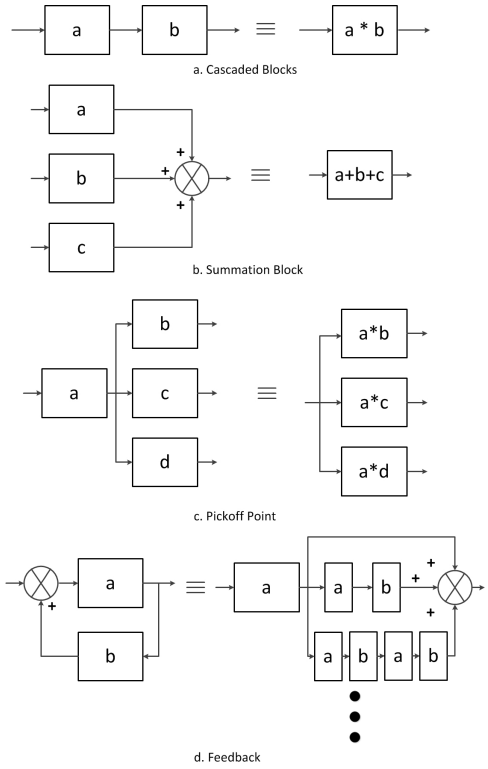

Fig. 1. Basic Building Blocks of Control System Block Diagram

Definition 5: `∀ a b.feedback a b = series [a; ∑∞k=0 branch a b k]`

The function `feedback` accepts the forward path transfer function `a` and the feedback path transfer function `b` and returns the net transfer function by forming the series network of the summation of all the possible infinite branches and the final forward path transfer function, since the output is taken after the forward path `a`.

In order to provide reasoning support for the above mentioned definitions, and thus minimize human intervention in formal verifying control systems, we verified some useful properties. A couple of key properties are mentioned here.

Firstly, we verified a key relationship for the feedback loop:

Theorem 1: *Feedback Loop Simplification Rule*

`∀ a b.|a * b|<1 ⇒feedback a b = a / (1 - a*b)]`

The complex normalization function is formalized in [8]. The proof of Theorem 1 is primarily based on the infinite summation of a geometric series [8]. This is a classical result and the net transfer function of a feedback block is always expressed in this form in all control system design texts [14], [15]. However, interestingly, the assumption of Theorem 1 is usually not stated in such texts, without which the proof does not hold. Missing such assumptions is a common problem in paper-and-pencil based mathematical analysis and it may lead to faulty system designs.

Similarly, we also verified the equivalence relationship between the block diagrams shown in Fig. 2.

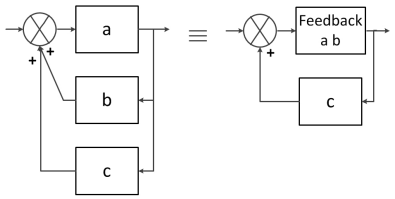

Fig. 2. Multiple Feedback Simplification Rule

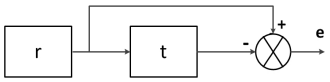

Fig. 3. Steady-State Error Model

Theorem 2: *Feedback loop simplification*

```
∀ a b c. |a*b| + |a*c| < 1 ∧ |a * b| < 1 ⇒
feedback a (sum_j (pickoff 1 [b;c])) =
    feedback (feedback a b) c
```

The proof of Theorem 2 involves Theorem 1 along with a rigorous complex arithmetic reasoning. But the usefulness of verifying such general universally quantified expressions is that they can be reused to simplify control system block diagrams to attain a net closed loop transfer function almost automatically. The net closed loop transfer function plays a vital role in reasoning about many characteristics of a control system, such as the stability, steady-state error and sensitivity.

III. Steady-State Error Analysis

Steady-state error analysis [14] is primarily used to find the deviation of system behavior from the desired course of action. The classical test inputs for the analysis of steady-state error are *step*, *ramp* and *parabola*. The block diagram corresponding to the steady-state error is shown in Fig. 3. Here r and t represent the predefined input and the net transfer function of the system under test, respectively. We can formalize the steady-state error as follows:

Definition 6:
```
∀ r t s. ss_error_bd r t s =
  sum_j (pickoff (r s) [1; -(t s)])
```

The function `ss_error` accepts the functions `r` and `t` and a complex number `s`. The negative sign with `(t s)` is used to cater for the subtraction in the summation junction.

The steady-state error is defined as the limit of the net transfer function of the block diagram, shown in Fig. 3, when time t approaches infinity. This limit, i.e., $\lim_{t\to\infty} f(t)$, corresponds to the limit ($\lim_{s\to 0} sf(s)$), in the Laplace domain [14]. This can be modeled as follows:

Definition 7: `∀ r t. ss_error r t =` $\lim_{s\to 0}$ `(s * (ss_error_bd r t s))`

We can now use the above definition to formally verify the steady-state errors corresponding to the three commonly used cases of step, ramp and parabola functions. The Laplace transforms of step, ramp and parabola functions are $\frac{1}{s}$, $\frac{1}{s^2}$ and $\frac{1}{s^3}$, respectively, and these expressions can be used to replace

the variable r of Definition 7 to verify the following theorems, where r represents the input to the system.

Theorem 3: *Steady-state Err. with Step, Ramp & Parabola Inputs*

```
∀ t. ss_error (λs.1/s) t = lim_{s→0} (1-t(s))
ss_error (λs.1/s^2) t = lim_{s→0} ((1-t(s))/s)
ss_error (λs.1/s^3) t = lim_{s→0} ((1-t(s))/s^2)
```

The proofs are based on the definition of limit of a function. The formalization presented so far in this paper consumed about 300 man-hours, which are mainly spent in the user guided verification due to the undecidable nature of the higher-order logic. Our proof script is available at [1]. The main benefit of this development, however, is that it greatly facilitates the formal reasoning about further control system properties by reducing the human interaction in such proofs, as will be illustrated in the next section.

IV. Application: Solar Tracking System

Solar tracking systems are used for the optimum generation of electricity from solar energy. These systems are usually deployed in aerospace applications where solar power is the only source for power generation. The block diagram representation of a traditional control system used in a solar tracking system is shown in Fig 4 [10], where k is gain adjustment factor. This system can be formalized, using our definitions, as follows:

Definition 8: *Solar Tracking System*

```
∀ s k. stcs s k = feedback
(series [(s + 0.01)/s );
    feedback (series [(s + 0.01)/s;
      k; 1/s]) (-1); 1/s]) (-1)
```

where inv(x) represents the inverse function $1/x$. The net transfer function of this system can be verified as the following theorem by utilizing the theorems described in Section II.

Theorem 4: *Net Transfer Function of Solar Tracking System*

```
∀ s k. |k * (s + 0.01)/s pow 2|< 1 ∧
|(k * (s + 0.01) pow 2)/
((s pow 2)*((s pow 2)+k*(s+0.01)))|< 1 ⇒
stcs s k =
 (k*((s + 0.01) pow 2))/
  ((s pow 2)*((s pow 2)+k*(s+0.01))+
    k*((s + 0.01) pow 2))
```

where the function x pow n denotes the power function x^n. The distinguishing features of the above theorem include its continuous and generic nature, due to the use of universal quantification over complex number variables, and its absolute correctness, due to the soundness of theorem proving.

The next step is to use this net transfer function to verify that the steady-state error of this solar tracking system is 0 for the step and ramp function inputs and $100/k$ for the parabolic function input using Theorem 3. The main challenge in the verification of this theorem was the rewriting of the term stcs s k based on Theorem 4 in the expression $lim_{s\to 0}$ (1 - stcs s k), where s is not a free variable. Thus, the expression had to be simplified using the definition of the function lim and we have to find the set of assumptions that bound the variable k such that the assumptions of Theorem 4 are satisfied in this context.

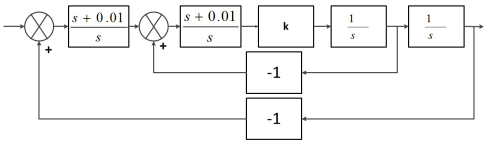

Fig. 4. Solar Tracking Control System

V. Conclusions

This paper presents a formal framework to reason about the net transfer functions of control systems, which play a vital role in assessing various interesting characteristic. We also provide the reasoning support for one such characteristic, i.e., the steady-state error. Our formalization is based on the complex number theory available in HOL-Light theorem prover and allows us to cater all kinds of control systems including their continuous elements. For illustration, we also presented some analysis of a real-world solar tracking control system. Some of the interesting future directions in this line of research is to formally analyze the stability of control systems and formalizing Laplace transform theory to be able to link time domain and Laplace domain models of a control system.

References

[1] M. Ahmad. Formal Analysis of Steady State Errors in Feedback Control Systems. http://save.seecs.nust.edu.pk/students/ahmad/sse.html, 2012.

[2] R. Alur. Formal Verification of Hybrid Systems. In *Embedded Software*, pages 273–278, 2011.

[3] R. Arthan, P. Caseley, C. O'Halloran, and A. Smith. ClawZ: Control Laws in Z. In *Formal Engineering Methods*, pages 169–176, 2000.

[4] R.J. Boulton, H. Gottliebsen, R. Hardy, T. Kelsey, and U. Martin. Design Verification for Control Engineering. In *Integrated Formal Methods*, volume 2999 of *LNCS*, pages 21–35, 2004.

[5] R.J. Boulton, R. Hardy, and U. Martin. A Hoare Logic for Single-Input Single-Output Continuous-Time Control Systems. In *Workshop on Hybrid Systems, Computation and Control*, volume 2623 of *LNCS*, pages 113–125, 2003.

[6] J. Harrison. *Theorem Proving with the Real Numbers*. Springer, 1998.

[7] J. Harrison. A hol theory of Euclidean Space. In *Theorem Proving in Higher Order Logics*, volume 3603 of *LNCS*, pages 114–129, 2005.

[8] J. Harrison. Formalizing Basic Complex Analysis. *Studies in Logic, Grammar and Rhetoric*, 10:151–165, 2007.

[9] H. Herencia-Zapana, R. Jobredeaux, S. Owre, P. Garoche, E. Feron, G. Perez, and P. Ascariz. PVS Linear Algebra Libraries for Verification of Control Software Algorithms in C/ACSL. In *NASA Formal Methods*, volume 7226 of *LNCS*, pages 147–161. Springer, 2012.

[10] R . R. Kumar, P. A. Cooper, and T. W. Lim. Sensitivity of Space Station Alpha Joint Robust Controller to Structural Modal Parameter Variations. *Journal of Guidance, Control, and Dynamics*, 15(6):1427–1433, 1992.

[11] M. D. Lutovac and D. V. Toic. Symbolic Analysis and Design of Control Systems using Mathematica. *International Journal of Control*, 79(11):1368–1381, 2006.

[12] B. Mahony. The DOVE approach to the Design of Complex Dynamic Processes. In *Workshop on Formalising Continuous Mathematics*, pages 167–187. NASA conference publication, 2002.

[13] MathWorks Simulink. www.mathworks.com/products/simulink, 2012.

[14] N.S. Nise. *Control System Engineering*. Wiley and Sons, 2003.

[15] K. Ogata. *Modern Control Engineering*. Prentice-Hall, 1997.

[16] L. Pike. Pervasive Formal Verification in Control System. In *Formal Methods in Computer-Aided Design*. Panel Discussion, 2011.

[17] A. Tiwari and G. Khanna. Series of Abstractions for Hybrid Automata. In *Workshop on Hybrid Systems: Computation and Control*, volume 2289 of *LNCS*, pages 465–478, 2002.