
Proceedings Paper:
Jasim, O.A. and Veres, S.M. (2017) Towards Formal Proofs of Feedback Control Theory. In: System Theory, Control and Computing (ICSTCC), 2017 21st International Conference on. 2017 21st International Conference on System Theory, Control and Computing (ICSTCC), 19-21 Oct 2017, Sinaia, Romania. IEEE, pp. 43-48. ISSN: 2372-1618.

Submitted to the ICSTCC 2017 Conference

Towards Formal Proofs of Feedback Control Theory

Omar A. Jasim
Department of Automatic Control and Systems Engineering
The University of Sheffield
Sheffield, UK
E-mail: oajasim1@sheffield.ac.uk

Sandor M. Veres
Department of Automatic Control and Systems Engineering
The University of Sheffield
Sheffield, UK
E-mail: s.veres@sheffield.ac.uk

***Abstract*—Control theory can establish properties of systems which hold with all signals within the system and hence cannot be proven by simulation. The most basic of such property is the stability of a control subsystem or the overall system. Other examples are statements on robust control performance in the face of dynamical uncertainties and disturbances in sensing and actuation. Until now these theories were developed and checked for their correctness by control scientist manually using their mathematical knowledge. With the emergence of formal methods, there is now the possibility to derive and prove robust control theory by symbolic computation on computers. There is a demand for this approach from industry for the verification of practical control systems with concrete numerical values where the applicability of a control theorem is specialised to an application with given numerical boundaries of parameter variations. The paper gives an overview of the challenges of the area and illustrates them on a computer-based formal proof of the Small-gain theorem and conclusions are drawn from these initial experiences.**

***Keywords*—Feedback Control; Robust Control; Small-Gain Theorem; Formal Methods.**

I. Introduction

The purpose of control design is to produce feedback, feedforward and adaptive controllers to provide robust performance in practical applications. In some industrial areas control performance needs to be guaranteed due to safety, economic or productivity requirements. These controllers need to be officially certified that they conform to standards. The analysis of controllers for certification has traditionally relied on symbolic computation. Such symbolic computation is not only algebraic but it needs the use of the concepts of signal spaces and nonlinear operators. Theorem provers, which are computer software that use mathematical symbols with the aid of some logical techniques, need to be able to handle nonlinear causal operators and prove properties of their interconnections into a feedback system as well. Higher-Order Logic (HOL) [1] is also needed because it includes quantifications and type theory such as real and complex numbers that make the implementation of control properties applicable. An example of such properties is the using of high-order functions to define nonlinear operators. With the advance of automated reasoning, such formal analysis can now enter the possibilities of control system design beside traditional methods of manual derivations.

Given the performance specifications, where specifications are the mathematical models of the desired properties, a control system is designed [2]. Then, code and electronics are developed and chosen. Fig.1 displays the three stages of formally verifiable controller design. Stage 1 is a precise mathematical definition of the plant (the system to be controlled), sensor and actuator dynamical variations and performance requirements, against which the implemented control system is to be verified. Stage 2 consists of the computer aided design (CAD) of a controller, which can be mathematically proven to perform up to specifications, the primary topic of this paper using computer based proofs. Stage 3 is the implementation of the mathematical model of the controller in computer code, which should not introduce bugs or numerical errors serious enough to make the specifications violated. Finally, the code should be free from bugs, which is ensured by code verification. In this paper we are interested in formal verification algorithms, which aims to check the correctness of control design (CAD) in Stage 2.

Often simulations are used to see whether the design is acceptable for the performance required. Simulations, however, may not uncover all signal combinations, which cause control performance to fail. By their definitions, robust CAD methods that rely on control theory will achieve performance requirements. Then the remaining problem is to prove that encoding does not make control performance underachieved due to computational errors in Stage 3.

The new formal verification methods proposed in this paper need to precede software verification of controller code as they verify the correctness of control algorithms, which are implemented in software. This paper gives a short overview of past use of formal methods to verify the correctness of control system implementations to place our work in context. We recall efforts made to formally verify that the code used in practice correctly implements the control algorithms intended. We also review methods of proving mathematical theorems by computers. None of these past works addresses the verification of the control theory and algorithms by formal methods in the form of symbolic computation to prove control theory on which the control algorithms are based. For reliability and safety of practical control systems, both algorithmic verification (to be introduced in this paper for the first time) and verification of controller implementation are needed (the latter

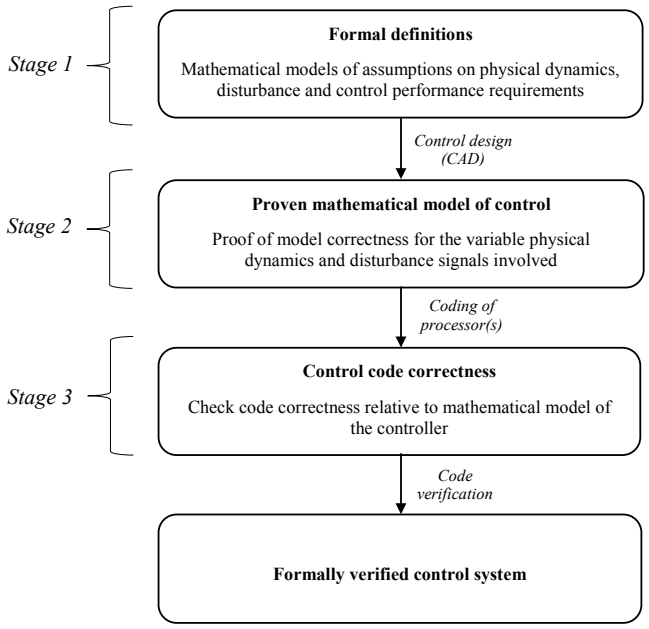

Fig. 1. The three principle stages which lead to practical control system verification.

pursued by many researchers in the past). This will provide high standards of certification in the future.

As a first attempt to prove control theory by a computer, we have chosen one of the most fundamental and general result of nonlinear feedback system, the "Small-gain theorem" (SGT). This is a fundamental theoretical result for many practical applications and plays an important role in robust control theory [3]. Through this first example, our aim in this paper is to describe the existing difficulties in the technical execution of formal proofs needed for control theory in the future.

In practice, both the plant and the feedback controller suffers from the variability of dynamics and disturbances. The Small-gain theorem can be used to assess feedback stability for plants with variable dynamics, for which norm bounds can be measured in experiments. If for all plant and controller variations the product of the norm of the plant and the controller is less than 1, then the feedback loop is robustly stable.

As no theory is yet widely known for automating the proofs of control systems, formal proofs in control theory, this paper intends to provide an initial approach to this challenge and gives an illustration on a formal proof of the Small-gain theorem using interactive theorem prover. The paper structure as follows: Section 2 introduces the research area of computer-based formal proofs for control theory. Section 3 illustrates the challenges and methods of the analysis and proof of the Small-gain theorem of robust feedback control. Section 4 illustrates the shortcomings of the available formal methods. Finally, Section 5 shows conclusions.

II. Formal Methods and Related Works

A. Formal Methods

Formal methods techniques or tools are based on mathematical logic and they are used for specification, design, and verification of hardware and software systems. The most useful technique to prove mathematical theories is theorem proving. There are two different essential techniques of theorem proving, Automated Theorem Proving (ATP) [4] which is automatically proving mathematical formulas by computer software, and Interactive Theorem Proving (ITP) or proof assistant which use to develop formal proofs by human-machine collaboration. There is also another proving type called Satisfiability Modulo Theories (SMT) [5], which is an extension of the Boolean satisfiability problem (SAT), that is a problem of deciding the satisfiability of first-order formulas in addition to some background theory with respect to some decidable first-order theory [6]. However, the difference between ITPs and ATPs is that the later are systems which include a set of decision procedures that use for automatic proof a specific restricted formats of mathematical formulas. Moreover, ATPs have limited expressivity where some mathematical theories cannot be stated and proved using them. In contrast, ITPs provide the ability to the user to formalize a mathematical theory and let the system prove it using already existing logical and mathematical expressions or theories, with the aid of some proof techniques and ATPs such as SPASS and Vampire or SMT solvers such as CVC4 and Z3.

Another well-known tool in formal methods is model checking, an automatic technique use for verifying finite state systems [7]. ITPs, in contrast to model checkers, can be applied to an infinite state space design. However, there are many other features of ITPs such as generality in terms of results and applicability. For producing results, which are general and represented as symbolic quantities that can be substituted by other quantities if the conditions are satisfied and types are matched. Regarding applicability, it is also general as the rules of logical deduction which are essentially general. In addition to modularity as each theory can be defined and then used or modified during theories formalizing and proving. Therefore, the total system is a comprehensive model of correlated theories, i.e., each theory can be built from other theories according to the relations and requirements.

B. Formal Methods in Control Theory

Typically, control systems design starts with formal analysis followed by numerical implementation in a simulation tool, then numerical simulations checking for valid behaviour before deploying the implementation. Recently, the use of autocoding generation techniques have increased that produce real-time code from the simulation which reduces manual coding errors. However, nowadays, complex control systems could be designed using digital computation techniques which have been rapidly developed in the last few decades. This enables systems to be formally checked and verified to ensure their validity and reliability. The outcome of that could be significant because system modelling using some mathematical

derivations can be checked and verified precisely using formal methods like proof assistants which ensure system robustness.

There is a wide range of ITPs such as Isabelle/HOL [8], Coq [9], PVS [10], which are HOL systems that can be used to verify the stability and performance of control systems with the aid of ATP like MetiTarski [11]. The current development of these techniques enables them to prove the most abstract robust control theories which are used to check systems stability. This work is motivated by the need of robust techniques for physical control systems validation and verification. It devotes to integrate control theory with ITP techniques by formally proving some of the most important theorems in control theory. We believe that this will be beneficial especially in safety-critical systems such as flight control, autopilot, autonomous cars and human interactive robots whereby systems stability and performance will be more robustness and safer. We also think that information from control theory can be translated into formal mathematical and logical concepts. These concepts then can be proved using proof assistants which can be used later in control systems verification. In particular, for complex systems where computations are very complicated and they are difficult to be handled by a human while it could be done by computer more easily and accurately. To prove that this is applicable, the Small-gain theorem is formally proved in Isabelle/HOL proof assistant.

Due to the importance of the verification of engineering systems in general and on control systems stability and performance using formal methods in particular, some related works in this area are mentioned. Hardy [12] evolved and performed a decision procedure to justify about a function that has a finite number of inflection points. This method carried out in the Nichols plot Requirements Verifier (NRV) to implement an automated formal Nichols plot analysis using computer algebra system (Maple) and PVS proof assistant in addition to other tools. NRV used to verify two control systems: an inverted pendulum and a disk drive reader. Akbarpour and Paulson [13] were also formally proved these two systems later using MetiTarski ATP also relying on Maple and Nichols plot analysis. In [14] authors presented an approach and tools to translate discrete-time Simulink models to the LESAR model checker. These tools have been applied to translate part of Audi's automotive controller. An extension of this work can be found in [15] where further analysis methods are introduced to define a subset of Stateflow for which synchronous semantics can be defined. In [16], Denman and his colleagues presented a method to implement formal Nichols plot analysis by using the MetiTarski automated theorem prover for stability verification. They extracted the transfer function of a flight control system from Simulink. Then, they defined an exclusion region of the Nichols Plot and proved the unreachability of the exclusion region using MetiTarski. Finally, they applied their proposed method into an autopilot model to check its validity.

Some verification processes can be done at the design level such as in SimCheck [17] where an implementation of type checking with custom annotations in Simulink blocks was presented. Similar work can be found in Araiza-Illan and her colleagues work [18] where they developed a new approach to automatic translating system's block diagrams modelled in Simulink into the Why3 [19] platform to verify their corresponding properties. The modelled system in Simulink represented high-level properties of stability (Lyapunov stability [20]), feedback gain and robustness. In [21], same authors presented a different approach by performing checking and comparing the results produced by a simulation through assertion checks and the results produced from the Why3 to determine the advantages of the latter. On the other hand, other verification processes can be accomplished at the code level such as in Feron work [22]. He developed a credible autocoder tool to produce target C code from Simulink that represent the system specifications in addition to documents that associated with the target code which represents properties of their proofs. Jobredeaux [23], proposed in his thesis an extension of [22] by a credible autocoding framework and tools used to develop the state of formal analysis of control software. The framework produced and proved high-level properties of control laws using PVS, such as closed-loop stability, in code level using the C code.

There have been many attempts made in the same direction of the previously presented works but using different methodologies and various formal methods such as in [24]–[26].

III. Formal Proof of the Small-Gain Theorem

A. Mathematical Proof of the Small-Gain Theorem

In order to prove the Small-gain theorem in a general way, we relied on the version and proof of the theorem presented in Khalil's book [27, Sec 5.4]. Our review has found that this version was one of the most general proofs using general nonlinear operators and stability concepts.

The following are the mathematical procedures of the proof, which we need to present first to enable us to comment on the respective steps of a computer-based proof procedure.

If we consider the relation of an input/output system as

$$y = Hu, \tag{1}$$

where $H : u \to y$ is an operator that maps the signal u onto the signal y. The input signal u belongs to a space of signal functions over the time interval $[0, \infty]$ into the Euclidean space R^m ($u : [0, \infty] \to R^m$). For the space of piecewise continuous, bounded and square integrable functions, the norm can be defined by

$$\|u\|_{L2} = \sqrt{\int_0^\infty u^T(t)u(t)dt} < \infty, \tag{2}$$

where the norm function, which is used to measure the size of the signal, should satisfy the following properties:

- $u = 0 \Longleftrightarrow \|u\| = 0$ else $\|u\| > 0$,
- $\|au\| = a\|u\|$ for $\forall a \in \Re$ and $a > 0$,
- $\|u_1 + u_2\| \leq \|u_1\| + \|u_2\|$.

We assumed that the input and output signals belong to the same space so that

$$L = \{u, y\ ,\ u_\tau, y_\tau| \quad \forall \tau \in [0, \infty]\}, \tag{3}$$

where L is a linear space and u_τ, y_τ are input and output truncated signals, respectively. The u_τ is a truncation of u that is defined by

$$u_\tau(t) = \begin{cases} u(t), \ 0 \leq t \leq \tau \\ 0, \qquad t > \tau \end{cases} \tag{4}$$

The proof required some definitions such as system's causality and stability, see [27, Sec. 5.1]. The causality property of an operator $H : L \to L$ is defined by $(Hu)_\tau = (Hu_\tau)_\tau$ for all $\tau \geq 0$. Using this property we can define the stability

$$\|(Hu)_\tau\| \leq \gamma \|u_\tau\| + \beta, \tag{5}$$

where $\gamma, \beta \in \Re$ and $\gamma, \beta > 0$, for all $u \in L$ and $\tau \in [0, \infty]$. For the proof of the SGT, suppose we have two systems $H_1 : L \to L$ and $H_2 : L \to L$, which are both finite-gain stable so that:

$$\|y_{1\tau}\| \leq \gamma_1 \|e_{1\tau}\| + \beta_1, \quad \forall e_1 \in L, \ \forall \tau \in [0, \infty), \tag{6}$$

$$\|y_{2\tau}\| \leq \gamma_2 \|e_{2\tau}\| + \beta_2, \quad \forall e_2 \in L, \ \forall \tau \in [0, \infty), \tag{7}$$

and we also assume that for each input $u_1, u_2 \in L$, there exist unique outputs $e_1, y_1, e_2, y_2 \in L$ where $u = [u_1 \quad u_2]^T$, $y = [y_1 \quad y_2]^T, e = [e_1 \quad e_2]^T$. The corresponding feedback system is illustrated in [27, Fig. 5.1].

Theorem: Under the above assumptions with finite gains γ_1 for H_1 and γ_2 for H_2, the feedback system is finite-gain stable if $\gamma_1 \gamma_2 < 1$.

Proof: Assuming existence of the solution, we can write

$$e_{1\tau} = u_{1\tau} - (H_2 e_2)_\tau, \quad e_{2\tau} = u_{2\tau} + (H_1 e_1)_\tau, \tag{8}$$

then,

$$\begin{aligned} \|e_{1\tau}\| &\leq \|u_{1\tau}\| + \|(H_2 e_2)_\tau\| \leq \|u_{1\tau}\| + \gamma_2 \|e_{2\tau}\| + \beta_2 \\ &\leq \|u_{1\tau}\| + \gamma_2 (\|u_{2\tau}\| + \gamma_1 \|e_{1\tau}\| + \beta_1) + \beta_2 \\ &= \gamma_1 \gamma_2 \|e_{1\tau}\| + (\|u_{1\tau}\| + \gamma_2 \|u_{2\tau}\| + \beta_2 + \gamma_2 \beta_1), \end{aligned} \tag{9}$$

since $\gamma_1 \gamma_2 < 1$,

$$\|e_{1\tau}\| \leq \frac{1}{1 - \gamma_1 \gamma_2} (\|u_{1\tau}\| + \gamma_2 \|u_{2\tau}\| + \beta_2 + \gamma_2 \beta_1). \tag{10}$$

$$\|e_{2\tau}\| \leq \frac{1}{1 - \gamma_1 \gamma_2} (\|u_{2\tau}\| + \gamma_1 \|u_{1\tau}\| + \beta_1 + \gamma_1 \beta_2). \tag{11}$$

for all $\tau \in [0, \infty)$. Finally, using the triangle inequality, we have

$$\|e\| \leq \|e_{1\tau}\| + \|e_{2\tau}\|. \tag{12}$$

B. Isabelle/HOL Overview

Isabelle is a generic interactive theorem prover which supports a variety of logics and provides interactive reasoning to prove formal mathematical theories or expressions using logical calculus. It is a specification and verification system written in the ML programming language [28] that represents rules as propositions (not as functions) and constructs proofs by combining rules that comprise a meta-logic based on lambda-calculus [29]. Isabelle provides useful proof procedures such as First-Order Logic (FOL), constructive type theory, Zermelo-Fraenkel set theory (ZF) [30], which offers a formulation of ZF on the top of FOL, and HOL.

The most common platform of Isabelle is Isabelle/HOL, which provides a higher-order logic theorem prover environment. Isabelle has a structured proof language called *Isar* in which proofs are conducted. *Isar* is a mathematics-like proof language that allows proofs to be easily readable and understandable for both users and computers.

Isabelle has been chosen by us due to its powerful logical techniques and its large library produced by a broad community of applied mathematicians. The most competitive alternative tool to Isabelle is Coq. The difference between them is minor from the technical point of view but Isabelle has more interesting and larger set of background theories in its library. For instance, Isabelle's library includes theorems ranging from logics, algebra and type theory such as HOL theory, reals, integers, complex numbers, and functions through spaces definitions such as topological spaces, Euclidian space, vector space and normed space to more complex theories such as derivative, integration, differential equations, high order functions, complex transcendental and operator norm. In addition to other features, for example, there is a code generation feature that allows to transfer the proven specifications from HOL syntax into a corresponding executable code in SML, OCaml, Haskell or the Scala programming languages [28], [29].

C. Formal Proof of the Small-Gain Theorem in Theorem Prover

To describe how SGT has been proved in Isabelle/HOL, this section will show the major steps of the proof procedures starting form definitions of time intervals, signals, truncations of signals, operators causality and stability. A signal's domain and range spaces are also declared in addition to a so-called truncation space and some properties and operations on signals which are also declared on these spaces. The definition of an operator space includes the declaration of their properties, which are defined in a general way to provide flexibility and re-usability for the development of other theories in the future.

In this work, some theories, which already exist and formally proven in Isabelle, have been exploited and used such as *"HOL.thy"*. HOL theory includes the axioms of logic in the higher-order form, the *"Multivariate Analysis.thy"*, which contains, for example, integrations, extended real and algebra theories, *"Bochner Integration.thy"*, which includes Lebesgue integration definition with their properties that are used in this work, *"set integral.thy"* which is used for the integration over a specific set or intervals, *"Function Algebras.thy"* that includes the properties of functions, for instance, pointwise addition, scalar multiplication, functions addition and multiplication, etc. As theories call other related theories automatically, there are several related theories called and used to carry through the proof of the theorem. All the codes that

we are formalized to prove the SGT can be found in our web-repository[1], as it is too long to be included in this paper.

The steps described previously in Section III.A are formalized in Isabelle as follows:

- **Time interval**: Before formalizing the theorem, some definitions are needed to be completed such as definition of time interval bounds. The overall *time interval* (T) is defined as a real set $[0, \infty)$ such that $t \in T$ where t is a real variable, that is $T = \{t \,|\, 0 \leq t < \infty\}$. The *truncation time interval* (T_τ) which is a subset of T and $\tau \in T_\tau$ is the period between 0 and τ, where τ is the truncation point, such that $T_\tau = \{\tau \in T \mid 0 \leq t \leq \tau\}$.
- **Signal bounds**: The signal value range is defined over $(-\infty, \infty)$ as $R = \{r| \ -\infty < r < \infty\}$.
- **Signal definition**: The first attempt to define input signals was by using ordered pair theory. It was soon discovered that Isabelle does not support working with a set theory (which defined under Isabelle/ZF platform) under Isabelle's HOL platform as they are two distinct approaches. Therefore, this work relied on the set theory axioms defined in the Isabelle/HOL theory. The problem is that set theory in HOL is abstracted from Zermelo-Fraenkel (ZF) theory, which is under FOL, and it does not support functions as ordered pairs. For example, the first trial to define a signal was as $u \subseteq (T \times R)$ and $(t, x) \in u$ where x belonged to the range set of the signal u. Therefore, here u is of a type $((real \times real)set)$ which means a set of real ordered pairs. However, the input signal then needs to be defined as a piecewise continuous function by the following general formula:

$$u : T \to R \,;\; u = (\forall t \in T, \exists^1 u(t) : u(t) \in R)$$

- **Domain and range space definition**: The domain and range spaces contain a set of signals, which are declared using "*locale*" feature in Isabelle which dealing with parametric theories. This feature enables us to form a definition with a set of assumptions in Isabelle. It is also gives a flexibility in dealing with spaces under certain constraints and properties and provides the possibility to add additional properties when the theory is called and used later. However, the domain space D and range space G have the same definitions and properties, each of which is defined as a set of signals (functions) under the properties of associativity, commutativity of addition, pointwise addition, distributivity of scalar multiplication and scalar multiplication over addition.
- **Signal truncation and truncation space definition**: The truncation of a signal is defined starting from declaring a definition that states what truncation means. It is represented as if there is an input signal u and there is a truncation point τ which belong to the interval $[0, \infty)$ such that all the values in the interval $[0, \tau)$ are valid and the values out of this interval are all set to zero. After that, truncation space TR is declared under specific constraints and all truncated signals should belong to this space.
- **Operator causality definition**: Because system stability is required for the proof of the SGT and from the fact that the system to be stable should be causal, system causality is defined. Causality is an important property of dynamical systems, which is needed to describe practical real-time feedback systems. A system is said to be causal if its output, $y(t)$, at any point depends only on its input, $u(t)$, up to that point. Therefore, with the truncation property the statement will be equivalent to $(Hu)_\tau = (Hu_\tau)_\tau$, which is easily stated in Isabelle.
- L_2 **norm - Cauchy-Schwarz and Minkowski integral inequalities**: Before defining system stability, there is a need to measure the norm of a signal with its specific properties. Because there is no norm definition in Isabelle/HOL that is suitable for SGT's proof, it was necessary for us to formalize and define a norm function. The norm function which should satisfy the properties mentioned in (2) is defined with the need to define Minkowski and Cauchy-Schwarz integral inequalities [31] to satisfy the required inequality property.
- **Input/output stability definition**: Input/output (I/O) stability is an essential aspect in the study of interconnected systems stability, where the increasing or decreasing nature of the signals norm can be tracked from the gain of the system. A system is said to be stable if it produces a bounded output for a bounded input. Therefore, I/O stability is an important part of the SGT. After completing the definitions of signals, truncation of signals, operators causality, and the norm function, it is possible to define the I/O stability as in (5).
- **Small-gain theorem formal proof**: After completing the required definitions for formalizing the proof of the theorem, it is possible now to apply the prove procedures step-by-step. The proof steps (8-12) can be applied in Isabelle/HOL under the same assumptions as in [27] in addition to other assumptions listed to perform the proof in Isabelle/HOL. Examples of such assumptions are signals u_1 and u_2 with their truncation, domain space, range space, truncation and operator spaces, causality and stability, and the integrable functions (signals). The proof steps need simple algebra, inequalities, substitutions and some arithmetic operations, which are proved in Isabelle/HOL platform.

IV. Shortcomings of available methods

Although Isabelle/HOL has an extensive list of proved theories, there was a need for more theories and formalizations to model control systems and their properties. Therefore, some theories and formulas were proved first before proving the SGT. The reason for this is that the library of Isabelle is still under development like other interactive theorem prover systems. For instance, the Cauchy-Schwarz's integral inequality, Minkowski's integral inequality and the norm of square integrable function are needed in the proof steps. Therefore, these

[1] *https://github.com/Formal-Methods-of-Robotics/Small-Gain-theorem*

theories in addition to some related lemmas are formalized and proved (See our web-repository). These theories have been proved by us because in the proof of the SGT, the norm with the integration of a function is needed and the norm definition that already exists in Isabelle library is not applicable. Also, formalizing and proving ZF over HOL platform are needed to work on signals and operators sets. These are just examples of the current limitations of ITPs for proving control theories. Other mathematical concepts are needed to formally prove such theories especially for those dealing with inequalities, which are considerably used in control theory. These concepts are utilized to formalize and prove control theory statements in the formal verification process.

Inequalities involving real-valued special functions are more effective to prove in the MetiTarski theorem prover but it has not yet been integrated with Isabelle/HOL. Moreover, we cannot easily use MetiTarski in association with Isabelle as it is an automated theorem prover (ATP). We have had to add theories to Isabelle to deal with control engineering problems. Examples of such improvements are by proving mathematical concepts related to control aspect such as inequalities, convergence concepts, norms, extending ordered-pair theory over HOL, improving set theory over HOL, function algebras, operators, operator norm, etc. Also, there is a need for a collaboration between computer scientists and control engineers to develop and extend theories in theorem proving to improve the formal verification process and this will ultimately lead to assure the robustness of control systems.

V. CONCLUSIONS

The work carried out so far has indicated that even the most theoretical control concepts involving nonlinear operators, causality and normed spaces of signals over the infinite semi-axis of time can be handled by formal languages and theorem proving techniques in higher-order logic using Isabelle/HOL and associated tools. The proof of the Small-gain theorem in Isabelle/HOL indicated that the highly abstract and general control systems can be handled by automated reasoning. We also found that there is a possibility to formulate and prove other control theories using ITPs. This may need to formalise some related mathematical concepts to prove the intended control theories. The ultimate aim of our research is to achieve practical industrial benefits of this emerging computational technology to support certification of the safety and quality of future control systems.

REFERENCES

[1] T. F. Melham, *Higher order logic and hardware verification*. Cambridge University Press, 2009, vol. 31.

[2] R. C. Dorf and R. H. Bishop, *Modern control systems*, 11st ed. New Jersey: Pearson Prentice-Hall, 2008.

[3] M. Green and D. J. Limebeer, *Linear robust control*. Courier Corporation, 2012.

[4] M. Fitting, *First-order logic and automated theorem proving*. Springer Science & Business Media, 2012.

[5] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability modulo theories." *Handbook of satisfiability*, vol. 185, pp. 825–885, 2009.

[6] R. Sebastiani, "Lazy satisfiability modulo theories," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 3, pp. 141–224, 2007.

[7] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. London: MIT press, 1999.

[8] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL: a proof assistant for higher-order logic*. Springer Science & Business Media, 2002, vol. 2283.

[9] G. K. G. Huet and C. Paulin-Mohring, "The Coq proof assistant: A tutorial," in *INRIA [Online]. Available: http://coq.inria.fr*. INRIA, 2016.

[10] S. Owre, J. M. Rushby, and N. Shankar, "PVS: A prototype verification system," in *International Conference on Automated Deduction*. Springer, 1992, pp. 748–752.

[11] L. C. Paulson, "Metitarski: Past and future," in *International Conference on Interactive Theorem Proving*. Springer, 2012, pp. 1–10.

[12] R. Hardy, "Formal methods for control engineering: A validated decision procedure for nichols plot analysis," Ph.D. dissertation, University of St Andrews, 2006.

[13] B. Akbarpour and L. C. Paulson, "Applications of MetiTarski in the verification of control and hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2009, pp. 1–15.

[14] P. Caspi, A. Curic, A. Maignan, C. Sofronis, and S. Tripakis, "Translating discrete-time Simulink to Lustre," in *International Workshop on Embedded Software*. Springer, 2003, pp. 84–99.

[15] N. Scaife, C. Sofronis, P. Caspi, S. Tripakis, and F. Maraninchi, "Defining and translating a "safe" subset of Simulink/Stateflow into Lustre," in *Proceedings of the 4th ACM International Conference on Embedded Software*, ser. EMSOFT '04. New York, NY, USA: ACM, 2004, pp. 259–268. [Online]. Available: http://doi.acm.org/10.1145/1017753.1017795

[16] W. Denman, M. H. Zaki, S. Tahar, and L. Rodrigues, "Towards flight control verification using automated theorem proving," in *NASA Formal Methods Symposium*. Springer, 2011, pp. 89–100.

[17] P. Roy and N. Shankar, "Simcheck: a contract type system for Simulink," *Innovations in Systems and Software Engineering*, vol. 7, no. 2, pp. 73–83, 2011.

[18] D. Araiza-Illan, K. Eder, and A. Richards, "Formal verification of control systems' properties with theorem proving," in *2014 UKACC International Conference on Control (CONTROL)*, July 2014, pp. 244–249.

[19] F. Bobot, J.-C. Filliâtre, C. Marché, G. Melquiond, and A. Paskevich, "The Why3 platform," *LRI, CNRS & Univ. Paris-Sud & INRIA Saclay, version 0.87.3 edition*, 2016.

[20] J.-J. E. Slotine, W. Li *et al.*, *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199, no. 1.

[21] D. Araiza-Illan, K. Eder, and A. Richards, "Verification of control systems implemented in Simulink with assertion checks and theorem proving: A case study," in *2015 European Control Conference (ECC)*, July 2015, pp. 2670–2675.

[22] E. Feron, "From control systems to control software," *IEEE Control Systems*, vol. 30, no. 6, pp. 50–71, 2010.

[23] R. J. Jobredeaux, "Formal verification of control software," Ph.D. dissertation, Georgia Institute of Technology, 2015.

[24] G. Brat, D. Bushnell, M. Davies, D. Giannakopoulou, F. Howar, and T. Kahsai, "Verifying the safety of a flight-critical system," in *International Symposium on Formal Methods*. Springer, 2015, pp. 308–324.

[25] R. J. Boulton, H. Gottliebsen, R. Hardy, T. Kelsey, and U. Martin, "Design verification for control engineering," in *International Conference on Integrated Formal Methods*. Springer, 2004, pp. 21–35.

[26] A. Cavalcanti and P. Clayton, "Verification of control systems using Circus," in *11th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'06)*, 2006, pp. 10 pp.–.

[27] H. K. Khalil, *Nonlinear Systems*. Prentice-Hall, New Jersey, 1996.

[28] R. Milner, *The definition of standard ML: revised*. MIT press, 1997.

[29] G. Michaelson, *An introduction to functional programming through lambda calculus*. Courier Corporation, 2011.

[30] A. A. Fraenkel, Y. Bar-Hillel, and A. Levy, *Foundations of set theory*. Elsevier, 1973, vol. 67.

[31] G. H. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities*. Cambridge university press, 1952.