Electronic Communications of the EASST
Volume 079 (2020)

Interactive Workshop on the Industrial Application of Verification and Testing ETAPS 2020 Workshop (InterAVT 2020)

Block-Based Models and Theorem Proving in Model-Based Development

Cinzia Bernardeschi, Andrea Domenici, Adriano Fagiolini, and Maurizio Palmieri

8 pages

Guest Editors: Stylianos Basagiannis, Goetz Botterweck, Anila Mjeda
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Block-Based Models and Theorem Proving in Model-Based Development

Cinzia Bernardeschi[1], Andrea Domenici[1], Adriano Fagiolini[2], and Maurizio Palmieri[1]

[1] name.surname@ing.unipi.it
Department of Information Engineering
University of Pisa, Italy

[2] adriano.fagiolini@unipa.it
Department of Engineering
University of Palermo, Italy

Abstract: This paper presents a methodology to integrate computer-assisted theorem proving into a standard workflow for model-based development that uses a block-based language as a modeling and simulation tool. The theorem prover provides confidence in the results of the analysis as it guides the developers towards a correct formalization of the system under development.

Keywords: Model-based development, theorem proving, Matlab, PVS

1 Introduction

Model-based development relies on system modeling and simulation, using textual and graphical modeling languages that are both *declarative*, as they produce abstract but detailed descriptions of a system, and *executable*, as they generate code to simulate the system. This approach is very productive since it allows developers to explore and validate different design choices, and detect errors and problems at early development stages. However, the levels of dependability afforded by simulation may fall short of the strict requirements that current and future systems must satisfy.

Formal methods are increasingly being employed to provide a solid mathematical foundation to claims of correctness and dependability of complex – often safety-critical – systems. Formal methods, however, rely on languages and tools often unfamiliar to developers and quite different from the commonly used environments.

This paper aims at contributing to the deployment of formal methods in a standard model-driven development process, using them along with traditional modeling and simulation. More precisely, the proposed workflow begins by modeling a system with the Matlab/Simulink environment [sim], using its symbolic computation capability to extract a mathematical definition of its properties, and then analyzing these properties with the *Prototype Verification System* (PVS) theorem-proving environment [ORS92]. For the sake of illustration, this approach is demonstrated on a very simple system, a single-axle robot vehicle. Clearly, the advantage of coupling theorem proving to modeling and simulation would show in much more complex scenarios.

PVS is an *interactive* theorem prover, i.e., a user writes a *theory* defining the problem to study (e.g., a system design) and then proves theorems and lemmas (*goals*) about the theory (e.g., statements about system properties), building a proof in a series of steps by invoking prover commands interactively. Each prover command applies one or more inference rules. A PVS theory contains declarations of types, variables, constants, and functions, and logical statements to be proved (theorems or lemmas) or taken as valid *a priori* (axioms). Readers are referred to the literature for information on the language and inference system (e.g., [ORS92, COR+95]) or its applications (e.g., [BD16, MTDB17, BDM18]).

2 Overview of the approach

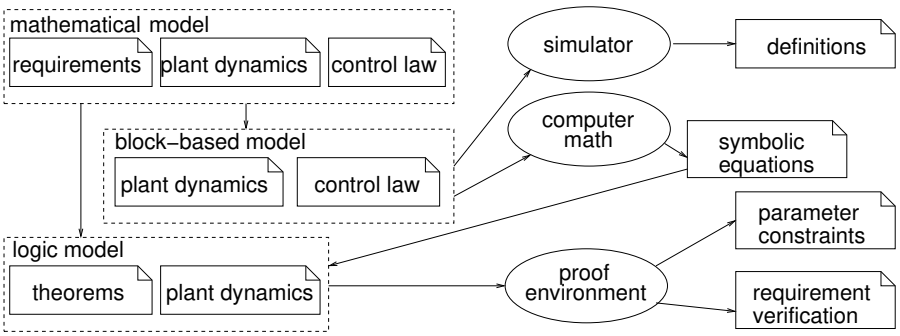

Figure 1: Computer-assisted mathematics and theorem proving.

The approach adopted in this paper is summarized in Fig. 1. The dynamical behavior of a system, its control law, and its requirements are initially described in terms of mathematical relationships. In a typical model-based process, the mathematical specification of plant and controller is expressed in a block-based model using such modeling and simulation tools such as Simulink or Modelica. The process exposed in this paper adds a step, where computer-assisted mathematics tools, such as the Symbolic Math Toolbox available in Matlab or Mathematica, are used to obtain symbolic solutions to relationships expressing requirements.

These symbolic solutions are included in a logic-based model, which represents the whole mathematical model, including the requirements, in the form of definitions and theorems in a higher-order logic language. Interactive theorem proving is then used both to verify requirements and to find constraints on design parameters that make requirements satisfiable.

It may be noticed that it is possible to limit the application of this process to specific components of the system, e.g., safety-critical ones. Also, not all components need a formal verification, such as parts that have been previously tested and verified.

3 A case study

This case study concerns the design of a controller for a robot vehicle. The robot must reach a given straight line l from an arbitrary initial position outside the line, as shown in Figure 2,

where d is the distance from l to the current robot position p, ψ is the current robot heading, and σ is the distance to the projection of p on l from a reference point σ_0. The robot must approach the line with a smooth curve, without oscillating. The robot moves on a flat surface without obstacles at a constant speed V and its turning speed ω is set by the controller.

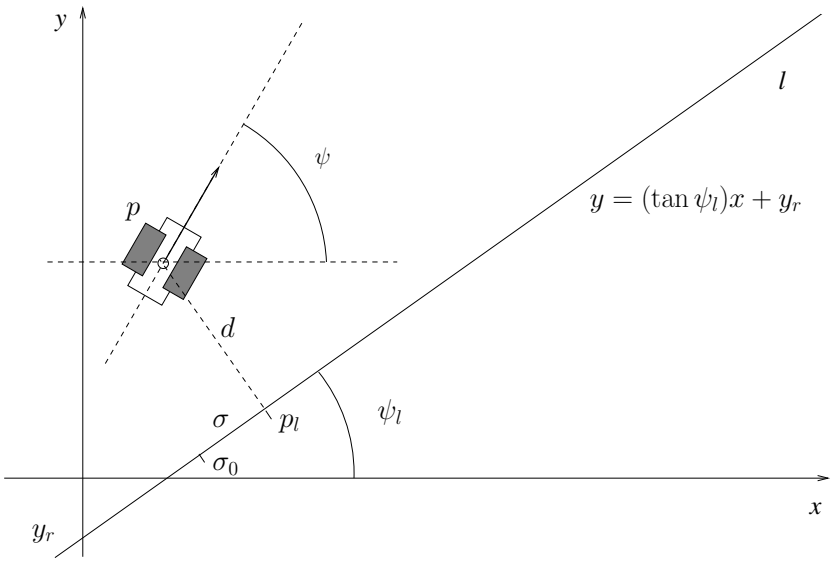

Figure 2: Line-seeking task for a robot vehicle.

With $\theta = \psi - \psi_l$, a control law is chosen with the form

$$\omega = -k_y d \mathrm{V} \operatorname{sinc} \theta - \mathrm{k}\theta \,, \tag{1}$$

where sinc equals $(\sin\theta)/\theta$ for $\theta \neq 0$ and 1 otherwise, and k is a design parameter ranging over the non-negative reals, whose value determines the stability and convergence properties of the system[1]. More precisely, it is necessary to find a range of values for k such that both d and θ approach zero as time approaches infinity, with the further requirement that the robot's trajectory does not oscillate.

The above task has been carried out with the following steps:

1. The system's generating functions have been found:

$$\begin{cases} \dot{\sigma} = \mathrm{V} \cos\theta \\ \dot{d} = \mathrm{V} \sin\theta \\ \dot{\theta} = -d\mathrm{V} \operatorname{sinc} \theta - \mathrm{k}\theta \,, \end{cases} \tag{2}$$

2. System (2) has been linearized with the Jacobian J, computed with the Matlab Symbolic

[1] Without loss of generality k_y can be chosen equal to $1\,\frac{rad}{m^2}$ and therefore it will be omitted in subsequent equations.

Math Toolbox (SMT) [smt], and evaluated for $d = 0$ and $\theta = 0$:

$$J_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & V \\ 0 & -V & -k \end{bmatrix} \tag{3}$$

3. The eigenvalues of J_0 have been computed with the SMT:

$$\lambda_1 = -\frac{\sqrt{k^2-4V^2}+k}{2} \qquad \lambda_2 = \frac{\sqrt{k^2-4V^2}-k}{2} \qquad \lambda_3 = 0\,. \tag{4}$$

4. The range of values for k that satisfies the requirements has been determined by interactive theorem proving, as discussed in the following subsection.

4 The Challenge

4.1 Parameter Synthesis by Theorem Proving

From control theory, the requirements presented above are fulfilled if the eigenvalues are real and nonpositive. After the control law has been chosen and the eigenvectors for the resulting linearized system have been symbolically computed, the remaining task is finding an appropriate range of values for the design parameter k. In this simple case, the task reduces to solving the following system for k, whose solution is $k > 2V$:

$$\sqrt{k^2 - 4V^2} \leq 0, \qquad \lambda_1 \leq 0, \qquad \lambda_2 \leq 0\,. \tag{5}$$

This simple system can be easily solved with the SMT (or, indeed, by paper and pencil), but it is worth considering how theorem proving can be used in this problem. The theorem-proving approach consists in defining the eigenvalues in a PVS theory and proving that a given range for k satisfies the requirements. This is shown more in detail in [DFP18].

4.2 Upper bound on the maximum distance of the vehicle from the line

In this section, theorem proving is used to find an upper bound to the distance of the robot from the target line, which would be hard to compute directly with such tools as SMT. Moreover, this analysis concerns a *discretized*, *executable* PVS theory, i.e., a theory written in such a form that a PVS interpreter [Muñ03] can simulate the system described by the theory, as it evolves in discrete time steps. An executable theory specifies a transition system whose states are sets of values for the system variables, and whose transitions are defined by a `step` function that maps the current state to the next one.

A simple example is considered, with the assumptions that the target line l is parallel to the x-axis, the initial heading ψ lies within the first quadrant, as shown in Figure 3, and parameter k of the control law is chosen within the region found above (Sec. 4.1).

The controller has been modeled as a discretized executable theory, defining the system state at a step i as a tuple $s_i = (\omega_i, x_i, y_i, \psi_i)$. At each step, the state is updated by the `step` function:

$$s_{i+1} = \texttt{step}(s_i)\,,$$

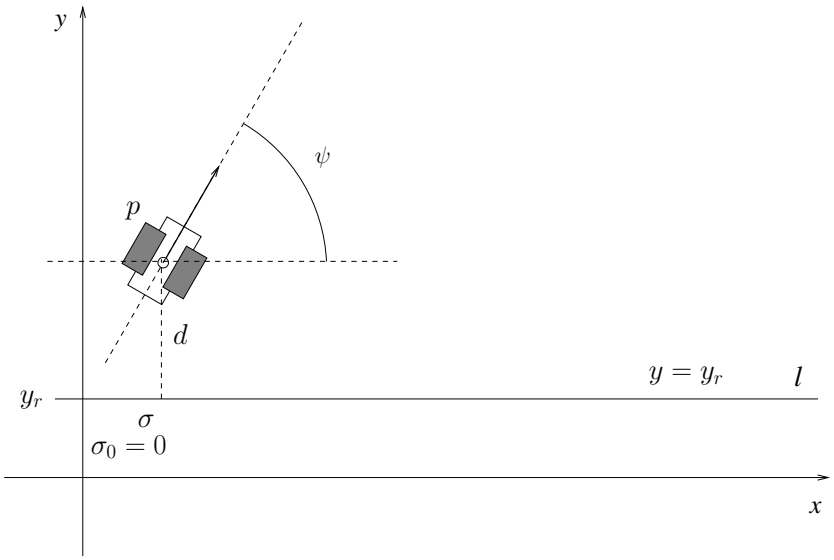

Figure 3: Line-seeking task for the line $y = y_r$.

which generates the sequence

$$\rho = s_0, s_1, \ldots, s_i, s_{i+1} \ldots .$$

Let the initial position and heading be $p_0 = (x_0, y_0)$ and $\psi = \psi_0$, respectively. The discretized equations are shown in (6), where $\Delta_t > 0$ is the discretization step and $i \geq 0$.

$$\begin{cases} x_{i+1} = x_i + \Delta_t V \cos\psi_i \\ y_{i+1} = y_i + \Delta_t V \sin\psi_i \\ \psi_{i+1} = \psi_i + \Delta_t \omega_i \\ \omega_i = -(y_i \cos\psi_l - x_i \sin\psi_l - y_r \cos\psi_l) V \dfrac{\sin(\psi_i - \psi_l)}{\psi_i - \psi_l} - k(\psi_i - \psi_l) \end{cases} \tag{6}$$

Since $\psi_l = 0$, the control law for ω reduces to (7):

$$\omega_i = -(y_i - y_r) V \frac{\sin\psi_i}{\psi_i} - k\psi_i \tag{7}$$

In the example of Figure 3, we take $x_0 = 2$, $y_0 = 4$, $0 \leq \psi_0 \leq \pi/2$, and $\omega_0 = 0$.

An upper bound on the distance d of the robot from l is found considering that (i) the vehicle departs from the line with $\psi > 0$, and (ii) ψ decreases at each step. From this, it follows that initially, the robot will move away from l, until ψ vanishes and the robot heads towards l. It is then possible to find an upper bound L on the increment of y at each step, and an upper bound N on the number of steps executed until ψ vanishes. As shown in Section 4.1, the condition $k > 2V$ guarantees that no further oscillations take place after that point. It is immediate to show that $\Delta_t V \sin\psi_0$ is the least upper bound for the y-increment (Lemma 5). An upper bound on d is then

$$d_{\text{ub}} = y_0 + N\Delta_t V \sin\psi_0 \,,$$

and we will focus on finding a value for N in the following.

We use an "instrumented" PVS theory, where the state is augmented with a Boolean variable α denoting that ψ has assumed values less than or equal to zero in the current state or in any previous state. In the initial state, since $\psi > 0$, $\alpha = \mathbf{F}$ (*false*). Then, when α becomes $\mathbf{T}$ (*true*), it remains equal to $\mathbf{T}$ forever. Equations (2) become (8):

$$\begin{cases} x_{i+1} = x_i + \Delta_t V \cos\psi_i \\ y_{i+1} = y_i + \Delta_t V \sin\psi_i \\ \psi_{i+1} = \psi_i + \Delta_t \omega_i \\ \omega_i(y_i, \psi_i) = -(y_i - y_r) V \dfrac{\sin\psi_i}{\psi_i} - k\psi_i \\ \alpha_{i+1} = \alpha_i \vee (\psi_{i+1} \leq 0) \end{cases} \tag{8}$$

The following lemma states some properties of ω and of the y co-ordinate of p in the next state. In particular, ω is negative, and the y co-ordinate is greater than y_r.

Lemma 1 *If $\psi_i \in (0, \frac{\pi}{2})$ and $y_i > y_r$ then $\omega_i < 0$ and $y_{i+1} > y_r$.*

Lemma 2 *Let $p_0 = (x_0, y_0)$ be the initial position, with $x_0 \geq 0$, $y_0 \geq 0$, and $\psi_0 > 0$, and let $y = y_r$ be the line equation, with $y_r < y_0$. Then,*

$$\forall_{i>0} \alpha_i \vee (\psi_i > 0 \wedge \psi_i < \psi_{i-1} \wedge y_i > y_r).$$

Lemma 3 *From the above hypotheses and lemmas, it follows that*

$$\forall_i (\forall_{j<i} (\alpha_j = \mathbf{F}) \wedge (\alpha_{i+1} = \mathbf{F})) \implies |\omega_i| > (y_0 - y_r) V \cos\psi_0).$$

For each step i such that $\alpha = \mathbf{F}$ for all preceding steps and $\alpha_{i+1} = \mathbf{F}$, the term $(y_0 - y_r) V \cos\psi_0$ is a lower bound to the decrement of ψ between two consecutive steps.

Lemma 4 *If j is such that $\alpha_j = \mathbf{T}$ and $\forall_{i<j} \alpha_i = \mathbf{F}$, and $N = \lceil \frac{\psi_0}{\Delta_t (y_0 - y_r) V \cos\psi_0} \rceil$ then $j \leq N$.*

N is an upper bound to the number of steps executed before ψ becomes less than or equal to zero.

Lemma 5 *We can prove that, for all states i such that $\alpha = \mathbf{F}$ for all preceding steps and $\alpha_{i+1} = \mathbf{F}$, the least upper bound of the y-increment is $\Delta_t V \sin\psi_0$:*

$$\forall_i (\forall_{j<i} (\alpha_j = \mathbf{F}) \wedge (\alpha_{i+1} = \mathbf{F})) \implies y_{i+1} \leq y_i + \Delta_t V \sin\psi_0 \, .$$

Theorem 1 *Let $d_{\max}$ be the maximum distance of the vehicle from the line. Then*

$$d_{\max} \leq y_0 + N \Delta_t V \sin\psi_0 \, .$$

4.3 Example of a Lemma Proof in PVS

In PVS, Lemma 3 is expressed with the following code:

```
lemma3 : LEMMA
 FORALL (i|i/=0):
  psi(i)>0 AND  psi(i)<pi/2 AND y(i)>y_r AND (
   FORALL(j|j<i): y(j+1)> y(j) AND psi(j+1) < psi(j))
   IMPLIES abs(w(i)) > (y(0)-y_r)*V*cos(psi(0))
```

The condition $\forall_{j<i}(\alpha_j = \mathbf{F}) \wedge (\alpha_{i+1} = \mathbf{F})$ is translated in PVS with the condition $\forall_{j<i}(y_{j+1} > y_j \wedge \psi_{j+1} < \psi_j)$. Using the command `(lemma lemma1)`, we can invoke Lemma 1, which states that $\omega_i < 0$, and then we can expand the `abs` function (command `(expand abs)`) and rewrite ω_i, obtaining the following *sequent*, where the AND of the negative-numbered formulas (*antecedents*) implies the positive-numbered one:

```
{-1}  w(i) < 0
{-2}  y(i+1) > y(i)
[-3]  psi(i) > 0
[-4]  psi(i) < pi/2
[-5]  y(i) > y_r
{-6}  FORALL (j: nat | j < i):
        y(j+1) > y(j) AND psi(j+1) < psi(j)
  |-------
{1}   k * psi(i) + sinc(psi(i)) * (y(i) - y_r) * V >
       (y(0) - y_r) * V * cos(psi(0))
```

Since $k\psi_i > 0$ and $V > 0$, we can simplify the consequent by removing $k\psi_i$ and canceling V (with the command `(cancel-by 1 "V")`), obtaining:

```
{-1}  w(i) < 0
{-2}  y(i+1) > y(i)
[-3]  psi(i) > 0
[-4]  psi(i) < pi/2
[-5]  y(i) > y_r
{-6}  FORALL (j: nat | j < i):
        y(j+1) > y(j) AND psi(j+1) < psi(j)
  |-------
{1}   sinc(psi(i)) * (y(i) - y_r) > cos(psi(0)) * (y(0) - y_r)
```

Knowing that $\forall_{\psi \in (0,\frac{\pi}{2})} \frac{\sin\psi}{\psi} > \cos\psi$, the previous consequent is true if $y_i > y_0 \wedge \text{sinc}(\psi_i) > \text{sinc}(\psi_0) > \cos\psi_0$. Using the antecedent `-6`, we can easily prove by induction (command `(induct j)`) that $y_i > y_0$ and $\psi_i < \psi_0$, which leads to $\text{sinc}(\psi_i) > \text{sinc}(\psi_0)$, and thus prove the consequent of Lemma 3.

5 Conclusions

Formal verification is important in the development of safety-critical systems. This work supports the thesis that computer-assisted verification can be integrated within model-based development of control systems. In particular, interactive theorem proving is used to verify a safety requirement of a nonlinear control system, not easily derivable by the Matlab/Simulink tools.

Possible future directions of this work should address (i) the proposal of a systematic, possibly automatic, translation from a subset of Simulink models to PVS theories, that will simplify the application of the proposed approach, (ii) the collection of proofs that can be reused to simplify the interactive verification process and integrate the theorem prover in the development process,

(iii) the application of the proposed approach to a more realistic case study in order to assess its feasibility. To this respect, this paper makes simplifying assumptions on perfect state measurement and exact knowledge of the system's model. More precisely, the typical assumption of slow-motion for the vehicle is made, implying that no lateral wheel slip occurs. In practice, the validity of such assumption depends on the underlying surface, not considered in this work.

Acknowledgments
The authors would like to thank the anonymous reviewers for their comments and suggestions. Work partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

Bibliography

[BD16] C. Bernardeschi, A. Domenici. Verifying safety properties of a nonlinear control by interactive theorem proving with the Prototype Verification System. *Information Processing Letters* 116(6):409–415, 2016.

[BDM18] C. Bernardeschi, A. Domenici, P. Masci. A PVS-Simulink Integrated Environment for Model-Based Analysis of Cyber-Physical Systems. *IEEE Transactions on Software Engineering* 44(6):512–533, 2018.

[COR+95] J. Crow, S. Owre, J. Rushby, N. Shankar, A. Srivas. A Tutorial Introduction to PVS. In *Proceedings of the Workshop on Industrial-Strenght Formal Specification Techniques (WIFT'95)*. April 1995.

[DFP18] A. Domenici, A. Fagiolini, M. Palmieri. Integrated Simulation and Formal Verification of a Simple Autonomous Vehicle. In Cerone and Roveri (eds.), *Software Engineering and Formal Methods*. Lecture Notes in Computer Science 10729, pp. 300–314. Springer International Publishing, Cham, 2018.

[MTDB17] G. Mauro, H. Thimbleby, A. Domenici, C. Bernardeschi. Extending a User Interface Prototyping Tool with Automatic MISRA C Code Generation. In *Proceedings of the Third Workshop on Formal Integrated Development Environment, F-IDE@FM 2016, Limassol, Cyprus, November 8, 2016.* Pp. 53–66. 2017.

[Muñ03] C. Muñoz. Rapid prototyping in PVS. Contractor report NASA/CR-2003-212418, NASA, Langley Research Center, Hampton VA 23681-2199, USA, May 2003.

[ORS92] S. Owre, J. Rushby, N. Shankar. PVS: A prototype verification system. In Kapur (ed.), *Automated Deduction — CADE-11*. Lecture Notes in Computer Science 607, pp. 748–752. Springer Berlin Heidelberg, 1992.

[sim] Simulink® web site.
http://www.mathworks.com/products/simulink

[smt] Matlab® SMT web site.
http://www.mathworks.com/products/symbolic