Constructing Verification Models of Nonlinear Simulink Systems via Syntactic Hybridization*

Nikolaos Kekatos[1], Marcelo Forets[1], and Goran Frehse[1]

Abstract— **In this paper, we present a methodology that facilitates the integration of formal verification techniques into model-based design. The focus is on set-based reachability analysis and on control systems that are described by hybrid dynamics and nonlinear components. Starting with a standard simulation model, e.g. in MATLAB/Simulink, we transform it into an equivalent verification model, formally a network of hybrid automata. This verification model complies with the SX format, which is a formalism used by several reachability tools. A major obstacle encountered is that highly scalable reachability algorithms and tools exist for piecewise affine (PWA) dynamical models, but not for nonlinear ones. To obtain PWA over-approximations of nonlinear dynamics, we use an abstraction method known as hybridization. Hybridization consists in partitioning the state-space into a set of domains and for each domain approximating the nonlinear dynamics by simpler ones. Nondeterministic inputs are added to account for the abstraction error. Existing hybridization procedures operate on the composed (flattened) system, so the number of partitions is exponential in the number of variables. This quickly leads to intractably large models, even for small systems. To mitigate this problem, we decompose the original dynamics and carry out the state-space partitioning and PWA approximation on the components. The number of partitions in each PWA component is at most quadratic in the abstraction error, thus largely avoiding an explosion in the number of partitions. Since the SX format can handle templates, several components may share the same abstraction. The result is a highly compact model that retains the modular structure of the original simulation model. If only a small subset of the partitions is reachable, the bottleneck of having excessively large PWA models can be avoided by composing the model on-the-fly during the reachability analysis.**

I. INTRODUCTION

In model-based design (MBD), the plant and its controllers are designed based on a model, typically within a simulation environment like MATLAB/Simulink [1], [2]. Any kind of nondeterminism in the system, like disturbances, measurement noise, parameter uncertainties, user input, or operating conditions, may have adverse effects on the performance. These effects can be difficult to predict during the design step. Therefore, the system is typically tested by simulating a large number of trajectories, each with a different choice for the nondeterministic quantities, and checking whether they satisfy the requirements. This process is generally incomplete since the number of different choices is prohibitively large or even infinite. Therefore, it can be hard to say with high confidence whether a requirement is truly satisfied under all circumstances. Formal verification attempts to guarantee that requirements are satisfied through a rigorous mathematical analysis of the system. A widely used verification technique is set-based reachability analysis, which exhaustively simulates families of trajectories using geometric operations on sets.

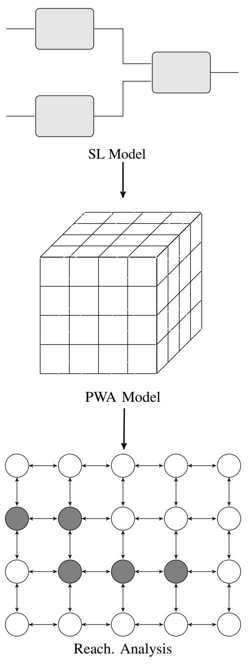
There are two main obstacles to applying reachability analysis in MBD. First, the simulation model needs to be converted to a suitable formal model, such as a hybrid automaton. Second, the model must be amenable to existing reachability algorithms, in particular in terms of scale. Highly scalable algorithms are known for piecewise affine (PWA) dynamical systems, but not for more complex nonlinearities. While a large class of nonlinearities can be approximated arbitrarily well by a PWA system, the resulting models can be very large, again running into scalability problems.

In this paper, we propose an approach to transform a simulation model into a compact, i.e. relatively small, verification model with PWA dynamics. To achieve this, we decompose the nonlinear system and perform the transformation component-wise. The resulting model can be fed to the verification tool SpaceEx [3] or translated into formats for other verification tools using the HyST tool [4]. Since SpaceEx composes the model on-the-fly during the analysis, only the reachable partitions of the PWA approximations are instantiated. Figure 1 illustrates the difference between the traditional hybridization methods and the proposed one, for the case where the original nonlinear model is described in Simulink. Classical hybridization techniques that rely on state-space partitioning [5] create a PWA model with $\mathcal{O}(1/\ell^n)$ locations, where ℓ is the mesh size, and n is the dimension of the state-space. During the reachability analysis, $\mathcal{O}(T/\delta)$ locations are visited, where δ is the minimum dwell time and T the global time horizon. On the contrary, with syntactic hybridization, we get m PWA components, where m is the number of nonlinearities, and the total number of locations is $\mathcal{O}(m/\ell^2)$. The non-reachable locations need not be instantiated.

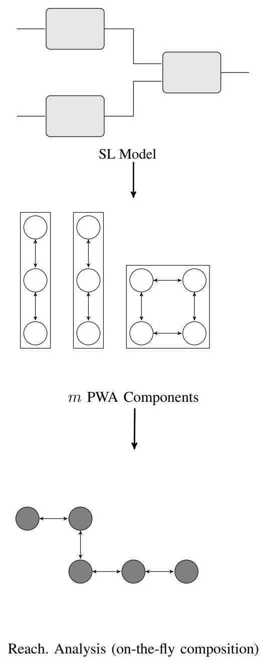
The rest of this paper is organized as follows. In Section II, we present the related literature. In Section III, we analyze the steps of our proposed methodology illustrating them with the use of a simple Simulink model example. In Section IV, we introduce the compositional syntactic hybridization and we apply it to an industrial case study. Finally, we draw conclusions and perspectives in Section V.

*This work was partially supported by the European Commission under grant 643921 (UnCoVerCPS) and by the Metro Grenoble through the project NANO2017.

[1]Nikolaos Kekatos, Marcelo Forets and Goran Frehse are with VERIMAG Laboratory, University Grenoble Alpes, Bâtiment IMAG, 700 Avenue Centrale, 38400 Saint-Martin-d'Hères, France. {Nikolaos.Kekatos, Marcelo.Forets-Irurtia, Goran.Frehse}@univ-grenoble-alpes.fr



(a) Traditional Hybridization



(b) Syntactic Hybridization

Fig. 1: Constructing verification models from a Simulink (SL) system, with $m = 3$ nonlinear blocks. The syntactic approach can lead to approximations of smaller size and less instantiated locations.

II. Related Work

The translation of Simulink models into modeling formalisms, for which formal verification tools can be applied, has attracted considerable interest; a comprehensive survey can be found at [6]. There exist translators from Simulink to Lustre [7], NuSMV model checker [8], and BIP [9]. However, all these tools apply discrete verification and do not consider continuous-time models. Filipovikj et al. [10] transformed Simulink models into the input language of UPPAAL Statistical Model checker. UPPAAL supports hybrid automata. However, it either restricts their continuous parts to simple dynamics or applies the Euler integration method. Its inaccurate integration results are therefore not conservative. Zuiliani et al. [11] presented a statistical model checking approach that is applicable to Simulink/Stateflow models. Stanley Bak et al. introduced a translation process from Simulink/Stateflow to hybrid automata in [12]. Both papers focus entirely on Stateflow diagrams and require the transformation of the Simulink model into a Stateflow one. However, this is not feasible for most large-scale systems designed with Simulink.

The translation of a Simulink/Stateflow model to a hybrid automaton is supported by the tools HyLink [12] and GreAT [13]. However, these tools do not allow hierarchical modeling and can be applied to a small subset of Simulink blocks. Recently, Minopoli and Frehse presented SL2SX, a semi-automated tool for translation of Simulink models into hybrid automata [14]. The translator supports a large number of Simulink blocks but is restricted by SpaceEx limitation to handle piecewise constant and affine dynamics. As a result, the user should analyze the missing blocks (unsupported or nonlinear) and decide how to replace or approximate them.

It is well-known that non-compositional methods for PWA approximations are not computationally efficient for complex systems, since an acceptable accuracy requires a very large number of pieces (locations) in the piecewise affine approximation [15]. Very recently, Deshmukh et al. [16] presented an experimental comparison of a compositional approach, similar to that presented in this paper (called nested approximation there), against a simplex-partitioning PWA hybridization, showing that the former scales much better than the latter for increasing demands on precision. The compositional PWA approximation is presented informally, while the paper neither discusses the implications in terms of the model size nor preserves this compositionality in the generated model. The complexity of the approximation can be further reduced by focusing on a set of reference trajectories, as done in [17].

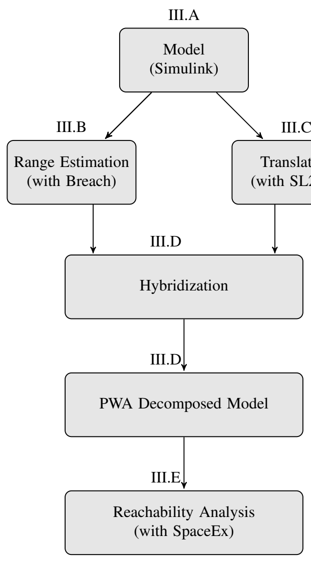
Much work has been done towards the verification of Simulink models [18], [6]. A promising group of approaches can be categorized as *verification by simulation* [19]. Donzé presented a MATLAB/Simulink based tool, *Breach*, which performs approximate reachability analysis and conducts efficient signal monitoring of properties and requirements. Breach facilitates the computation and property investigation of large sets of trajectories, but it still cannot provide absolute confidence in the simulation results. Another MATLAB toolbox that is designed to be seamlessly integrated into the model based design process of MATLAB/Simulink is S-Taliro [20]. S-Taliro conducts fast and efficient simulations but intrinsically relies on gridding, restricting the formal focus on falsification. The MATLAB/Simulink-based tool C2E2 [21] generalizes simulation trajectories to families of trajectories by deriving a neighborhood around the simulated trajectory in which all trajectories have equivalent behavior.

All the above verification by simulation approaches have in common that the set of initial states must be sampled. Since the number of required samples can increase exponentially with the number of state variables, this can limit the approach to systems with low-dimensional initial states. Also, simulation-based verification techniques can be used to verify properties for a bounded-time horizon, but cannot be applied for unbounded time. The tool HySon [22] performs set-based simulation directly on a Simulink model and computes a good approximation of the set of all possible executions. However, the technical details suggest that it may have its drawbacks when analyzing hybrid systems for an unbounded switching horizon. In addition, HySon is not publicly available.

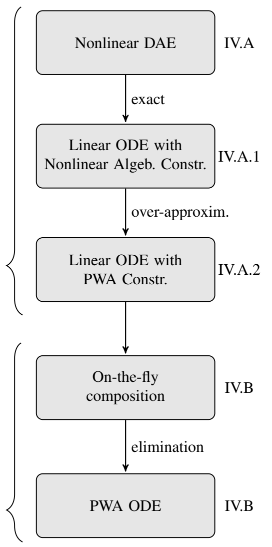
The main contribution of this work is to introduce a compositional syntactic hybridization method. This method is suitable for Simulink models, facilitates the construction of verification models, and takes advantage of the on-the-fly composition of hybrid systems that is supported by the SpaceEx platform.

III. Constructing Verification Models

In this section, we present the steps of our approach (portrayed in Fig. 2). To clearly illustrate our methodology, the proposed steps are applied to a rotational pendulum model.



(a) Verification Workflow



(b) Compositional Syntactic Hybridization Scheme

Fig. 2: Methodology for constructing verification models; input: Simulink, output: PWA decomposed model (left). The syntactic hybridization steps are presented (right). Each step (in Latin) is analyzed in the following sections.

A. Model-based design with Simulink

Modeling and control design are undertaken with MATLAB/Simulink, through the interconnection of blocks, signals, and systems. Simulink [2] is a graphical programming environment for modeling, simulating, and analyzing dynamical systems. It enables hierarchical modeling, keeping functionally related models together and simplifying the overall design process by means of abstraction.

Example III.1 *As a running example, we consider a simple rotational pendulum [23]. The pendulum has a nonlinear term (a sine function) and its Simulink model is shown in Fig 3. The system produces simulation traces of the pendulum angle over time, when it is released from rest.*

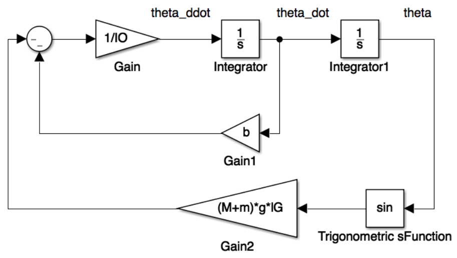


Fig. 3: Simulink model for the rotational pendulum

Simulink relies on *must* semantics, also known as urgent or as-soon-as-possible (ASAP) semantics. That means that discrete events/transitions occur as soon as a given condition (guard) is satisfied. On the other hand, most formal tools for reachability analysis use *may* semantics, demonstrating a broader set of behaviors. For the purposes of this paper, we consider that SL2SX takes care of these semantic differences [24].

B. Estimation of the signal range

In this step, the goal is to get bounds on the behavior (min, max) of the input signals of the Simulink blocks that cannot be described by linear or hybrid dynamics. The smaller the ranges of the signals, the smaller the number of locations that is required by the PWA abstraction, given a desired error bound. There are different ways to estimate them, such as simulations, interval analysis, or Monte Carlo methods. In this paper, we use the Breach [19] toolbox for a (not necessarily conservative) estimation of the signal range.

Note that the signal range serves only as an indication for the hybridization step that is presented in the next section. The approximation is equipped with out-of-range scopes. So, in case the range is shown to be insufficient during reachability computations, it is revised (enlarged).

Example III.2 *For the rotational pendulum, we estimate the range of the signal that acts as an input in the nonlinear block. The range is then enlarged by a percentage. A set of simulations for uncertain initial conditions and a Sobol distribution (quasi-random number sequence) are shown in Fig. 4, where we plot the angle θ as a function of time.*

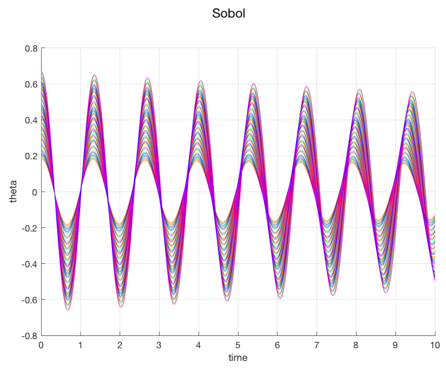


Fig. 4: Range estimation of the input signal of the nonlinear block (sin) of the rotational pendulum θ (rad) vs. time (s); conducted with Breach simulations.

C. Translation to SX format

The next step is to translate the Simulink model into an equivalent SpaceEx [3] model. SpaceEx models respect the semantics of SX grammar; the format is similar to the standard hybrid automata, syntactically extended with hierarchy and templates. Formally, a SpaceEx model is the tuple $SX = \langle Comp, Bind \rangle$, where $Comp$ represents the *components* (base or network) and $Bind$ is a relation that associates each network component with a set of components. A *base component* corresponds to a single hybrid automaton, whereas a *network component* corresponds to the parallel composition of several hybrid automata.

In the context of this work, we use the SL2SX [14] translator to handle the mechanical, but error-prone, aspects of deriving a hybrid automaton interpreted by SpaceEx from a Simulink model. The translator accepts a Simulink model that is saved in XML format and generates as an output a network of hybrid automata in SX format. The translation preserves most of the structural aspects of the Simulink diagram, such as the names, hierarchy, and graphical positions. The tool accepts several continuous-time, logical and arithmetical blocks, as well as blocks with discontinuous dynamics, such as switches.

Example III.3 *Applying the SL2SX translator to the Simulink model of the rotational pendulum, we obtain a SpaceEx model with base and network components. The top-level network component is shown in 5. The generated model can be easily compared with the Simulink diagram, due to the preservation of the structure and the names of blocks and variables. We have highlighted in red the nonlinear block, which is considered in the next subsection.*

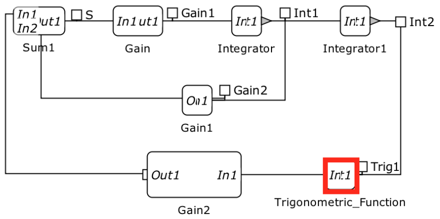


Fig. 5: SpaceEx model (SX) of the rotational pendulum constructed by the translator. This is the network component, and we highlight, in red, the block which corresponds to a trigonometric function.

D. Hybridization

This step aims to generate PWA approximations for the Simulink blocks that are not handled automatically by the translator, either because no exact translation is available (e.g. nonlinearities) or because translation cannot be applied (e.g. Embedded MATLAB Function). Recall that the PWA representation is a special class of hybrid automata admitting both discrete events, i.e. jumps, switches, and continuous dynamics in the form of piecewise affine functions.

After computing PWA approximations for the nonlinear Simulink blocks, we integrate them with the original XML file (constructed in the previous step). In this way, we get the complete SpaceEx model in SX format, combining the exactly translated blocks from SL2SX and the over-approximated blocks from syntactic hybridization. The resulting model can then be fed into the SpaceEx verification platform.

Example III.4 *As for the rotational pendulum, the nonlinear function* $(\sin)$ *of the rotational pendulum is over-approximated by a PWA function. We consider the linearization domains to be boxes and, for a given error bound, we get 40 domains. The approximation errors are computed for each domain (location in the resulting hybrid automaton) and are added in the form of a nondeterministic input in the invariants. The constructed SpaceEx base component is shown in Fig. 6.*

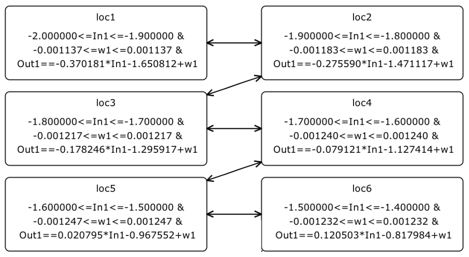

loc1
-2.000000<=In1<=-1.900000 &
-0.001137<=w1<=0.001137 &
Out1==-0.370181*In1-1.650812+w1
loc2
-1.900000<=In1<=-1.800000 &
-0.001183<=w1<=0.001183 &
Out1==-0.275590*In1-1.471117+w1
loc3
-1.800000<=In1<=-1.700000 &
-0.001217<=w1<=0.001217 &
Out1==-0.178246*In1-1.295917+w1
loc4
-1.700000<=In1<=-1.600000 &
-0.001240<=w1<=0.001240 &
Out1==-0.079121*In1-1.127414+w1
loc5
-1.600000<=In1<=-1.500000 &
-0.001247<=w1<=0.001247 &
Out1==0.020795*In1-0.967552+w1
loc6
-1.500000<=In1<=-1.400000 &
-0.001232<=w1<=0.001232 &
Out1==0.120503*In1-0.817984+w1

Fig. 6: PWA of the nonlinear component $(\sin)$ of the rotational pendulum. Only 6 locations are shown (out of 40). Here, the nondeterministic input w_1 represents the approximation error.

E. Reachability analysis with SpaceEx

The reachability analysis is undertaken by one of the SpaceEx analysis algorithms [3], i.e. STC, LGG, or PHAVer. SpaceEx composes the individual components on-the-fly, instantiating only the part of the model that is relevant. SpaceEx supports safety verification problems, albeit it is also possible to check richer properties or control specifications. Some traditional control objectives are encoded as reachability problems in [25].

Example III.5 *Computing the reachable sets of the rotational pendulum (for the STC SpaceEx scenario, a flowpipe tolerance of 0.01, a global time horizon of 1s), we get the phase portrait shown in Fig. 7. The pendulum is released from the most upward position with an uncertain but bounded initial speed.*

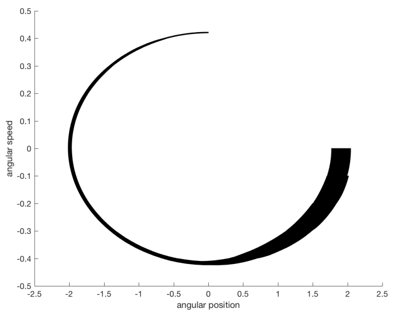


Fig. 7: Reachability results of the rotational pendulum for a global time horizon of 1s. Phase portrait of the angular position (rad) and angular speed (rad/s).

IV. COMPOSITIONAL SYNTACTIC HYBRIDIZATION

The objective of the compositional syntactic hybridization is to approximate the original model by a hybrid automaton with PWA dynamics in a compositional manner. Three main steps are involved: *syntactic decomposition*, replacing the original system by an equivalent one with extra variables; *hybridization*, constructing a PWA approximation for each domain and generating a sound over-approximation of the original system by adding an error term; and finally *HA composition*, where the PWA model is transformed into a hybrid automaton in SX format. The two first steps are introduced in Section IV-A, where the mathematical details are presented. The third step, which is implementation dependent, is presented in Section IV-B.

A. Syntactic PWA Aprroximation

In the following paragraphs, we explain the technical details involved, considering a nonlinear differential equation

$$\frac{dx}{dt} = f(x), \qquad x \in \mathbb{R}^n. \tag{1a}$$

This ODE is assumed to be regular (f is Lipschitz of constant $L > 0$ over the state-space $\mathcal{X} \subset \mathbb{R}^n$). The method can be extended to semi-explicit differential-algebraic equations (DAEs).

1) Syntactic Decomposition: The decomposition consists in constructing a new system where nonlinear terms are replaced by auxiliary variables,

$$\frac{dx}{dt} = g(x,y), \qquad y \in \mathbb{R}^m, \tag{2a}$$

$$y = h(x,y). \tag{2b}$$

Here y is a vector of auxiliary variables, $g(x,y) \in \mathbb{R}^n$ is linear in both x and y, and $h(x,y) \in \mathbb{R}^m$ includes all the nonlinear terms, m, of the original system, as explained in detail below. Notice that we have replaced the original system by a linear ODE in a higher-dimensional space, $\mathbb{R}^{n+m}$, coupled with a set of nonlinear algebraic constraints. Moreover, this step is exact.

Furthermore, let $V_i \subseteq \{x_1, \dots, x_n\}$ for $i \in \{1, \dots, m\}$ be the variables involved in the i-th nonlinearity, and let $p_i = |V_i|$ denote the number of variables in such expression. It should be noted that with a sufficient number of auxiliary variables, we can assume that $h_i(x,y)$ satisfies $1 \leq p_i \leq 2$ for all i.

2) PWA Approximation: We consider a set of *domains*, R_{ij}, which cover the operational range of the variables in V_i, where j is a label for each individual domain. For each R_{ij}, we perform a PWA linearization of h_i. Hence, (2a)-(2b) is replaced by

$$\frac{dx}{dt} = g(x,y), \qquad y \in \mathbb{R}^m, \tag{3a}$$

$$y = \hat{h}(x,y), \tag{3b}$$

where $\hat{h}$ is a vector of PWA functions.

Let op denote the operating point in the domain R_{ij}. Using Taylor's formula with the Lagrange remainder, for each $1 \leq i \leq m$,

$$\hat{h}_i(x,y) = h_{i,op} + \frac{\partial h_i}{\partial x}\Big|_{op}(x - x_{op}) + \frac{\partial h_i}{\partial y}\Big|_{op}(y - y_{op}), \tag{4}$$

and

$$h_i(x,y) - \hat{h}_i(x,y) = \frac{1}{2}(x - x_{op})^{\mathrm{T}}\frac{\partial^2 h_i}{\partial x^2}\Big|_{\xi}(x - x_{op}) + \tag{5}$$
$$\frac{1}{2}(y - y_{op})^{\mathrm{T}}\frac{\partial^2 h_i}{\partial y^2}\Big|_{\xi}(y - y_{op}) + (x - x_{op})^{\mathrm{T}}\frac{\partial^2 h_i}{\partial x \partial y}\Big|_{\xi}(y - y_{op}),$$

where $\xi = (\xi_x, \xi_y) \in \mathbb{R}^{n+m}$ is an intermediate point in the interval $\xi_x \in \{x_{op} + a(x - x_{op}), a \in [0,1]\}$, and similarly for ξ_y. The right-hand side of Eq. (5) is the Lagrange remainder, whose resulting values over the domain R_i are used to estimate the approximation error [26]. The linearization errors ϵ_h are computed by evaluation of the Lagrange remainder and satisfy $y = h(x,y) \in \hat{h}(x,y) \oplus \mathcal{B}\epsilon_h$, where $\mathcal{B}$ is the unit ball in the chosen norm $||\cdot||$. In this paper, we assume that R_i are boxes. In the case of a box, the point which minimizes the absolute value of the Lagrange remainder is its center [27]. Several interesting alternatives exist, such as simplices [5].

Example IV.1 *For $n = 4$, consider the polynomial vector field $f = [x_1 - x_2x_3x_4, x_1x_2 - x_4, -x_3x_4, x_2 - x_3]$. Introducing the auxiliary variables $y_1 = x_3x_4$, $y_2 = x_1x_2$ and $y_3 = x_2y_1$, then $g(x,y) = [x_1 - y_3, y_2 - x_4, -y_1, x_2 - x_3]$ is a linear ODE and $h(x,y) = [x_3x_4, x_1x_2, x_2y_1]$ is a nonlinear algebraic equation of degree two with $m = 3$ elements. Consider a PWA approximation, $\tilde{f}$, based on a rectangular partitioning of the state-space, with elements of size ℓ in each dimension. Then, $\tilde{f}$ leads to $O(1/\ell^4)$ elements, while the PWA approximation of h only to $O(m/\ell^2)$ elements.*

Example IV.2 *Now, we consider a 9-dimensional genetic model adapted from the one presented in [28], [29]. The model is described by polynomial dynamics of the form $f(x) = [3x_3 - x_1x_6, x_4 - x_2x_6, x_1x_6 - 3x_3, x_2x_6 - x_4, 3x_3 + 5x_1 - x_5, 5x_5 + 3x_3 + x_4 - x_6(x_1 + x_2 + 2x_8 + 1), 5x_4 + x_2 - 0.5x_7, 5x_7 - 2x_6x_8 + x_9 - 0.2x_8, 2x_6x_8 - x_9]$. Introducing the auxiliary variables $y_1 = x_1x_6$, $y_2 = x_2x_6$, $y_3 = x_6x_8$, then $g(x,y) = [3x_3 - y_1, x_4 - y_2, y_1 - 3x_3, y_2 - x_4, 3x_3 + 5x_1 - x_5, 5x_5 + 3x_3 + x_4 - y_1 - y_2 - 2y_3 - x_6, 5x_4 + x_2 - 0.5x_7, 5x_7 - 2y_3 + x_9 - 0.2x_8, 2y_3 - x_9]$ is a linear ODE and $h(x,y) = [x_1x_6, x_2x_6, x_6x_8]$ is a nonlinear algebraic equation of degree two with $m = 3$ elements. Consider a PWA approximation, $\tilde{f}$, based on a rectangular partitioning of the state-space, with elements of size ℓ in each dimension. Then, $\tilde{f}$ leads to $O(1/\ell^9)$ elements, while the PWA approximation of h only to $O(m/\ell^2)$ elements. Instead of gridding a 9-dimensional state-space, we only have to approximate with PWA functions three 2-dimensional state-spaces. This example highlights the usability of the syntactic approach for the cases that repeated nonlinearities appear.*

B. Compositional Hybridization

Through the syntactic PWA approximation, we have produced a Linear ODE with PWA algebraic constraints. In order to feed this model to one of the available reachability tools, we must describe it as a network of hybrid automata. Each hybrid automaton corresponds to the PWA approximation of one nonlinearity. Each piece of the PWA approximation corresponds to one location in the corresponding automaton.

In the SX file format, used by SpaceEx and other tools, a model consists of components which are either hybrid automata or networks of hybrid automata. A component can be instantiated inside a network, possibly remapping variables to other variables or replacing them with constant values. Note that an ODE or an algebraic constraint can be trivially embedded in a hybrid automaton with a single location. The ODE becomes the flow-constraint of the location and the algebraic constraint its invariant.

Expressing the PWA approximation in this setting, the linear ODE is modeled by a single (trivial) automaton. Each PWA constraint $y_i = \hat{h}_i(x, y)$ corresponds to a hybrid automaton with one location per piece. The locations of adjacent pieces are connected through transitions. The approximation error is expressed by extra variables with range ϵ_h, and the error threshold $\mu > 0$ is an upper bound on the maximum value it can take (in some chosen norm $||\cdot||$).

Example IV.3 *Getting back to Example IV.2, we model the genetic system with Simulink (see Fig. 8). The nonlinearities correspond to products and are highlighted with different colors. Blocks with the same color represent the same nonlinear operator.*

After translating the Simulink model and estimating the signal ranges, we are ready to perform syntactic hybridization on the components. As already mentioned in Section IV-A, we do not need to partition a 9-dimensional space, but only to construct 3 two-dimensional PWA approximations for the products. Note that the approximations correspond to base components in SpaceEx and can be seen as templates (building blocks). In this respect, it is possible to be reused. As a result, only one SpaceEx base component would be required to describe the three red blocks, one for the blue blocks, and another one for the yellow blocks.

Standard algorithms for reachability analysis, such as those inside SpaceEx, take as input a single hybrid automaton with ODE dynamics. To get from the multi-component input model to this form, the reachability tool performs two operations. First, it combines the components through a process called *parallel composition*. Second, it eliminates the algebraic constraints to obtain an ODE. In principle, parallel composition means building the product automaton, whose locations consist of the cross-product of the locations of the components. As such, a model with m components of k locations each has a product automaton with m^k locations. However, tools such as SpaceEx construct the product automaton on-the-fly, instantiating only the reachable locations.

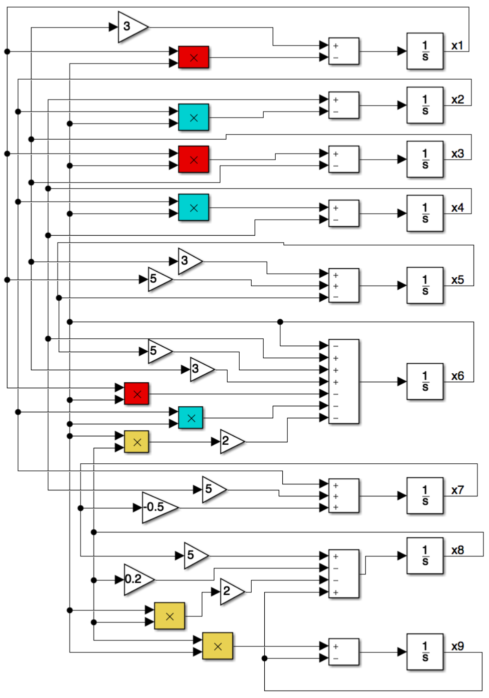


Fig. 8: Genetic model in Simulink. Colored blocks correspond to nonlinearities. Red: x_1x_6, cyan: x_2x_6 and yellow: x_6x_8.

Similarly, the conversion from linear DAE to ODE is only carried out on the instantiated locations. The conversion can be carried out efficiently by Gauss-Jordan elimination [30]. The underlying theory is explained in [31].

Example IV.4 *(continuation of example IV.3.) Conducting reachability analysis with SpaceEx for a global time horizon of 1s, with a flowpipe tolerance of 0.01 and the STC scenario (octagonal directions), we compute the reachable sets, as illustrated in Fig. 9.*

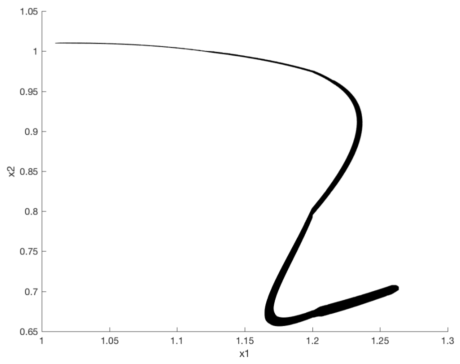


Fig. 9: Reachable sets of the genetic model with SpaceEx. The variables x_1 and x_2 are displayed.

In principle, the outlined procedure enables us to approximate the reachable set of the original dynamics with arbitrary precision. Let $\Phi(t,x)$ denote the trajectory starting from x evaluated at time t. The reachable set of the system from a set of initial points $X_0 \subseteq \mathcal{X}$ during the interval $[0,t]$ is defined as

$$Reach(t, X_0) = \{y = \Phi(\tau, x) : \tau \in [0,t],\ x \in X_0\}. \quad (6)$$

The approximate system converges to the original system, as expressed by the following theorem.

Theorem 1 *(see [32]) The Hausdorff distance between the reachable set of* (2a)-(2b) *and the reachable set computed through hybridization,* (3a)-(3b)*, from time* 0 *to a final time* $T > 0$ *satisfies*

$$d_H\big(Reach_f(T, X_0), Reach_{\hat{f}}(T, X_0)\big) \leq \frac{2\mu}{L}\big(e^{LT} - 1\big), \quad (7)$$

where μ is the error threshold, L is the Lipschitz constant of the original, nonlinear function, and $\hat{f}$ is the PWA approximation.

C. Case Study: Wind Turbine

The wind turbine benchmark from the ARCH workshop poses a challenging and relevant industrial model [33]. It is designed with MATLAB/Simulink and it is a large-scale model with many nonlinearities. We used syntactic hybridization in [34] to transform the model into an approximative PWA one in SX format and we conducted the approximation component-wise. Ten nonlinear blocks were approximated and Table I presents the corresponding results. The resulting model only has 72 locations in all components combined. The standard hybridization would lead to an $\mathcal{O}(1/\ell^n)$ number of locations. Taking into account that the dimension of the state-space is seven and considering 5 locations per state variable, the standard method would yield $5^7 = 78125$ locations. That indicates a considerable difference between the hybridization methods in terms of the model size.

V. Conclusions

Model transformation plays an important role in bridging the gap between industrially relevant models and verification tools [18]. This work aims to assist the application of hybrid system reachability tools to models designed with MATLAB/Simulink. We propose a methodology to construct verification models out of Simulink systems. We make use of the SL2SX translator to handle the mechanical aspects of the translation to hybrid automata and of the Breach toolbox to get bounds on the signal ranges. For the blocks that are not exactly translated, e.g. nonlinearities, we apply a new hybridization method, which we call compositional syntactic hybridization.

Unlike standard state-space hybridization methods, we do not operate over the fully composed (flattened) model but perform the PWA approximations component-wise. In this way, we can obtain a significant reduction in terms of model size. The constructed verification model consists of a network of hybrid automata and is described in the SX format. It can then be fed into the SpaceEx platform or other verification tools through the HYST translator. Using SpaceEx for the reachability computations, we can take advantage of the on-the-fly composition and instantiate only the reachable parts of the approximation. Note that our compositional hybridization can be applied not only to the dynamics but also to algebraic and initial constraints.

No	Block Type	# Loc	Error Bounds	Info
1	Product (2D)	4	2.00	$x \cdot y$
2	Division (2D)	10	2.76e-2	x/y
3	Division (1D)	2	2.01e-2	$\frac{1}{1+x}$
4	Division (2D)	4	6.98e-1	x/y
5	Product (2D)	4	25.6	$x^2 \cdot y$
6	Product (2D)	4	3.23	$x^2 \cdot y$
7	Polynomial (2D)	6	3.39	4th-order
8	Polynomial (2D)	6	12.3	4th-order
9	Embedded MATLAB	28	10.5	x^2 and $1/x$
10	Saturation	3	-	exact
11	Read from workspace	1	-	aux. variable
12	Mux, Scope	-	-	-
13	Save to workspace	-	-	-
14	Enabled Subsystem	-	-	-
15	Multiport Switch	-	-	-
16	Compare to Constant	-	-	-
17	Manual Switch	-	-	-

TABLE I: Wind Turbine Benchmark – breakdown into blocks. Blocks, 1-9, are approximated syntactically; the 10th block is exactly translated; 11th is replaced by a nondeterministic input and the rest, 12-17, are not necessary due to semantic differences.

On an industrial benchmark, the wind turbine, our approach leads to a very compact model that is orders of magnitude smaller than a standard hybridization model.

The next step is to improve the reachability tools so that they can efficiently employ these compositional models. The primary objective is to instantiate as few locations and transitions as possible during the analysis. There are three issues to address. The first is the on-the-fly composition and instantiation of the models, which can reduce the number of instantiated locations of the product automaton. The second is a compositional pre-processing of the components, where we utilize the pre-image of the target invariant when checking which transitions are enabled. The third direction would be to perform compositional mapping of the initial states. In the case of SpaceEx platform, the identification of the initial conditions is done through enumeration. However, enumeration of the locations of the product automaton is an operation that does not scale. Applying compositional reasoning would allow us to identify the initial locations and instantiate as few locations as possible.

Acknowledgement

The authors would like to thank Willem Hagemann and Stefano Minopoli for valuable discussions.

References

[1] G. Nicolescu and P. J. Mosterman, *Model-based design for embedded systems*. CRC Press, 2009.

[2] MATLAB 9.0 and Simulink 8.7, "The MathWorks, Inc., Natick, Massachusetts, United States."

[3] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2011, pp. 379–395.

[4] S. Bak, S. Bogomolov, and T. T. Johnson, "HYST: a source transformation and translation tool for hybrid automaton models," in *[HSCC'15]*, 2015, pp. 128–133.

[5] E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Informatica*, vol. 43, no. 7, pp. 451–476, 2007.

[6] N. Zhan, S. Wang, and H. Zhao, *Formal Verification of Simulink/Stateflow Diagrams: A Deductive Approach*. Springer International Publishing, 2016.

[7] S. Tripakis, C. Sofronis, P. Caspi, and A. Curic, "Translating discrete-time Simulink to Lustre," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 4, no. 4, pp. 779–818, 2005.

[8] B. Meenakshi, A. Bhatnagar, and S. Roy, "Tool for translating Simulink models into input language of a model checker," in *International Conference on Formal Engineering Methods*. Springer, 2006, pp. 606–620.

[9] V. Sfyrla, G. Tsiligiannis, I. Safaka, M. Bozga, and J. Sifakis, "Compositional translation of Simulink models into synchronous BIP," in *Industrial Embedded Systems (SIES), 2010 International Symposium on*. IEEE, 2010, pp. 217–220.

[10] P. Filipovikj, N. Mahmud, R. Marinescu, C. Seceleanu, O. Ljungkrantz, and H. Lönn, "Simulink to UPPAAL statistical model checker: Analyzing automotive industrial systems," in *FM 2016: Formal Methods: 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings 21*. Springer, 2016, pp. 748–756.

[11] P. Zuliani, A. Platzer, and E. M. Clarke, "Bayesian statistical model checking with application to Stateflow/Simulink verification," *Formal Methods in System Design*, vol. 43, no. 2, pp. 338–367, 2013.

[12] K. Manamcheri, S. Mitra, S. Bak, and M. Caccamo, "A step towards verification and synthesis from Simulink/Stateflow models," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*. ACM, 2011, pp. 317–318.

[13] A. Agrawal, G. Simon, and G. Karsai, "Semantic translation of Simulink/Stateflow models to hybrid automata using graph transformations," *Electronic Notes in Theoretical Computer Science*, vol. 109, pp. 43–56, 2004.

[14] S. Minopoli and G. Frehse, "SL2SX translator: From Simulink to SpaceEx models," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. ACM, 2016, pp. 93–98.

[15] G. Frehse, "Compositional verification of hybrid systems using simulation relations," Ph.D. dissertation, Radboud University Nijmegen, 2005.

[16] J. V. Deshmukh, H. Ito, X. Jin, J. Kapinski, K. Butts, J. Gerhard, B. Samadi, K. Walker, and Y. Xie, "Piecewise-affine approximations for a powertrain control verification benchmark," in *Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[17] S. Bak, S. Bogomolov, T. A. Henzinger, T. T. Johnson, and P. Prakash, "Scalable static hybridization methods for analysis of nonlinear systems," in *HSCC'16*, 2016, pp. 155–164.

[18] R. Alur, A. Kanade, S. Ramesh, and K. Shashidhar, "Symbolic analysis for improving simulation coverage of Simulink/Stateflow models," in *Proceedings of the 8th ACM international conference on Embedded software*. ACM, 2008, pp. 89–98.

[19] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 167–170.

[20] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-TaLiRo: A tool for temporal logic falsification for hybrid systems," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2011, pp. 254–257.

[21] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala, "Automatic reachability analysis for nonlinear hybrid models with C2E2," in *CAV'16*, 2016, pp. 531–538.

[22] O. Bouissou, S. Mimram, and A. Chapoutot, "HySon: Set-based simulation of hybrid systems," in *2012 23rd IEEE International Symposium on Rapid System Prototyping (RSP)*. IEEE, 2012, pp. 79–85.

[23] R. Tedrake, "Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for MIT 6.832," 2009.

[24] S. Minopoli and G. Frehse, "From simulation models to hybrid automata using urgency and relaxation," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. ACM, 2016, pp. 287–296.

[25] N. Loukkas, N. Meslem, and J.-J. Martinez-Molina, "An interval technique to check the performance of control laws applied to wind turbines," *SWIM 2016*, 2016.

[26] M. Berz and G. Hoffstätter, "Computation and application of Taylor polynomials with interval remainder bounds," *Reliable Computing*, vol. 4, no. 1, pp. 83–97, 1998.

[27] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 4042–4048.

[28] E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach, *Systems biology in practice: concepts, implementation and application*. John Wiley & Sons, 2008.

[29] X. Chen and S. Sankaranarayanan, "Decomposed reachability analysis for nonlinear systems," in *Real-Time Systems Symposium (RTSS)*, 2016.

[30] A. Donzé and G. Frehse, "Modular, hierarchical models of control systems in SpaceEx," in *Proc. European Control Conf. (ECC'13)*, Zurich, Switzerland, 2013.

[31] M. Gerdin, "Identification and estimation for models described by Differential-Algebraic Equations," Ph.D. dissertation, Institutionen för systemteknik, 2006.

[32] E. Asarin, T. Dang, and A. Girard, "Reachability analysis of nonlinear systems using conservative approximation," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2003, pp. 20–35.

[33] S. Schuler, F. D. Adegas, and A. Anta, "Hybrid modelling of a wind turbine (benchmark proposal)," Applied Verification for Continuous and Hybrid Systems (ARCH), 2016.

[34] N. Kekatos, M. Forets, and G. Frehse, "Modeling the wind turbine benchmark with PWA hybrid automata," in *4th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH)*, 2017.