Simulation of Hybrid Systems from Natural-Language Requirements

Bruno Oliveira[1], Gustavo Carvalho[1], Mohammad Reza Mousavi[2], Augusto Sampaio[1]

***Abstract*— Cyber-physical systems are characterised by a massive and tight interaction between computer systems and physical components. Hybrid systems provide an abstraction for modelling cyber-physical systems by featuring the integration of discrete and continuous behavioural aspects. Simulation is an important tool for validating hybrid system models, which are often too complex to be treated using other validation and verification techniques. Motivated by the industrial need for such tools, we propose a strategy (h-NAT2TEST) for simulation of hybrid systems from natural-language requirements. Using the proposed approach, one writes the system specification using a controlled natural language, from which an informal semantics is automatically inferred based on the case grammar theory. Then, a formal representation is built considering a model of hybrid data-flow reactive systems (h-DFRS). Finally, in order to allow for rigorous simulation, an Acumen specification is derived from the h-DFRS model. Simulation is supported by the Acumen modelling environment. A DC-DC boost converter is used as a case study to illustrate the overall approach.**

***Index Terms*— Hybrid systems; controlled natural language; data-flow reactive system; simulation; Acumen; NAT2TEST.**

I. INTRODUCTION

Cyber-physical systems (CPSs) have become ubiquitous in the industry and society, appearing in critical application domains such as aerospace, automotive, healthcare, and process control. Hybrid systems provide an abstraction to model the discrete and the continuous dynamics of CPSs that are often tightly intertwined aspects of CPS behavior [1].

Advances in technology continuously lead to the construction of increasingly complex CPSs; therefore, early and efficient validation and verification of CPSs is an eminent and timely challenge. Simulation is a lightweight approach to understanding and evaluating the behaviour of these systems. Simulation offers several advantages: firstly, it is a time-efficient technique. A computer simulation reflects the effects of real-world reactions and changes in a compressed time frame. Secondly, it is cost effective. It allows for early validation of engineering design without producing a physical prototype. Such an early validation is particularly valuable when prototyping is expensive (both in time and cost) and later removal of faults will be extremely costly [2].

Given the importance of hybrid systems modelling, simulation of such systems can bring about the above-mentioned benefits to the analysis of CPSs. Simulation for purely dynamical and purely discrete systems is well understood. There exist several numerical simulation methods for systems of Ordinary Differential Equations (ODE). However, the combination of discrete and continuous dynamics leads to challenging problems for simulation. The motivation for hybrid system simulation led to the development of several notations, tools and frameworks, such as 20-sim [3], Modelica [4], and Acumen [5], among others.

In this work our major and distinguishing goal has been to develop an environment that supports as early as possible simulations, generated directly from system-level requirements, and thus even before models are available. We base our solution (h-NAT2TEST) on a previous framework NATural Language Requirements to TEST Cases (NAT2TEST) [6], which automatically generates test cases from natural-language requirements of timed discrete systems. We extend NAT2TEST to hybrid systems leading to h-NAT2TEST. Although h-NAT2TEST is based on our earlier framework, supporting simulation of hybrid systems has a vertical impact on the original framework and, thus, all of its constituent phases were revisited. Our solution includes an extension of the original natural language to allow for the specification of continuous dynamics. As a consequence, more elaborate syntactic and semantic analyses were necessary, as well as an internal representation for requirements that combines both discrete and continuous behaviour. We translate this internal representation to Acumen and use the simulation environment of Acumen, which provides a rigorous analysis environment for hybrid systems [5]. To exemplify the proposed simulation strategy, we consider the requirements of a DC-DC boost converter and detail all the automatic steps until we obtain a simulation in Acumen.

Section II presents the background to this work. Section III explains how the NAT2TEST strategy is extended to allow for simulation of hybrid systems. Section IV details the application of our proposal to specify and simulate a DC-DC boost converter. Section V discusses related work. Section VI summarises our results and comments on future work.

II. BACKGROUND

We first recall basic concepts of hybrid systems in Section II-A considering a running example: a DC-DC boost converter [7]. Afterwards, in Section II-B, we describe the original NAT2TEST strategy, which is adapted here to allow for simulation of hybrid systems.

A. Hybrid Systems

Hybrid systems feature a combination of discrete and continuous events. These events coexist, interact and change in response to dynamics described by differential or difference equations in time [1]. One way to define the behaviour of

[1]`Universidade Federal de Pernambuco, Centro de Informática, 50740-560, Brazil, {bmo,ghpc,acas}@cin.ufpe.br`
[2]`Center for Research on Embedded Systems, Halmstad University, Sweden, m.r.mousavi@hh.se`

hybrid systems is via the set of all possible trajectories of the variables associated with the system. In this context, a hybrid system is represented as a *hybrid automaton model*.

A hybrid automaton consists of a finite-state automaton extended with continuous variables. The discrete changes are modelled by edges between the locations of the automaton (possibly guarded by expressions over continuous variables). The continuous evolution is modelled by differential equations that are associated with these locations [8].

A classical example of a hybrid system is a DC-DC Boost Converter (BC): it raises the voltage while decreasing the current from its input to its output. This type of device is widely used in modern equipment. For example, the engines used in electric vehicles, which require much higher voltage than the one that can be provided by a single battery.

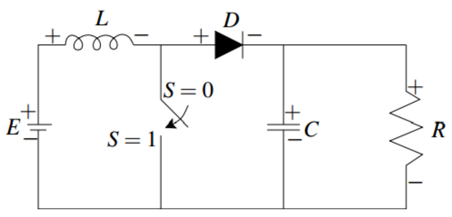

Fig. 1. Electrical circuit of a boost converter

Fig. 1, extracted from [7], shows an electrical circuit characterising the physical elements of this system. This boost converter is composed of an Inductor (L), a capacitor (C), a switch (S), a diode (D), and a resistive load (R); E denotes the input voltage. Closing S generates a short circuit from the right-hand side of L to the negative input. Consequently, a current flows from the positive to the negative supply terminals through L, which stores energy in its magnetic field. Opening S causes this accumulated energy to flow to the right side of the circuit, where it charges C. When S is closed again, energy is provided to L by C, which is recharged each time the switch opens again. It is desired to maintain an almost steady output voltage across L. Although the electrical circuit of the boost converter is a continuous system, the control system turns it into a hybrid one.

The control system of the boost converter has four operating modes according to the current position of the switch and whether the diode D is conducting/blocking energy. In each mode, the continuous evolution of the electric charge of the capacitor (q) and the magnetic flux of the inductor (Φ) is defined by two differential equations, respectively. For instance, when the switch is open and the magnetic flux of the inductor is positive, the continuous evolution of q and Φ is defined as $q' = \frac{\Phi}{L} - \frac{1}{RC}q$ and $\Phi' = -\frac{1}{C}q + E$, respectively. Therefore, the behaviour of this system can be formally modelled as a hybrid automaton, which has four different locations (one for each operating mode) connected by the appropriate edges and with two differential equations mapped to each location.

B. NAT2TEST

NAT2TEST [6] is an entirely automatic strategy for test case generation from natural-language requirements that was developed in close contact with industry. This strategy aims at reactive systems, whose behaviour can be described as actions that should be taken when certain conditions are met. For an automatic processing of requirements and generation of test cases, the requirements must be written according to a specific grammar, namely the System Requirements Controlled Natural Language (SysReq-CNL).

The SysReq-CNL is a Controlled Natural Language (CNL) grammar that has been created in order to turn the writing of system specifications more standardised, facilitating the conversion of the system requirements into a formal notation. In particular, this grammar supports writing requirements of data-flow reactive systems: a class of embedded systems where inputs and outputs are always available as signals.

After writing the system requirements according to the SysReq-CNL grammar, informal semantics is given to the specification by means of the case grammar theory [9]. In this theory, a sentence is analysed in terms of the semantic (thematic) roles (TR) played by each word in the sentence. The verb is the main element of the sentence, and it determines the role that each word plays with respect to the action or state described by the verb. The verbs' associated TRs are aggregated into a case frame (CF). Each verb in a requirement specification gives rise to a different CF. All derived CFs are joined afterwards to compose what we call a requirement frame (RF). In other words, the semantics of each system requirement is given by the corresponding RF.

The RFs are used to derive a formal representation of the system behaviour: a model of data-flow reactive system (DFRS [10]). This formal representation is a symbolic, timed and state-rich automata-based notation for representing natural-language requirements. Afterwards, for generating test cases, different notations and tools can be used and, thus, the DFRS model is accordingly translated into the chosen formalism. Fig. 2 shows the phases of this approach. The three initial phases are fixed: (1) syntactic analysis, (2) semantic analysis, and (3) DFRS generation; the others are related to the chosen formalism to generate test cases.

III. NAT2TEST for Hybrid Systems

The main purpose here is to support simulation of hybrid systems from system-level requirements, since more abstract requirements (e.g., stakeholder requirements) are not necessarily feasible for direct simulation. To allow for simulation, the requirements are translated to an Acumen [5] model and its environment is used for simulation. As illustrated in Fig. 2, our extension (h-NAT2TEST) impacts all fixed stages of the original NAT2TEST strategy. In what follows, we describe how each stage is properly adapted. More details can be found in [11].

A. Syntactic Analysis

The SysReq-CNL grammar is extended conservatively to accommodate requirements for hybrid systems. The dynam-

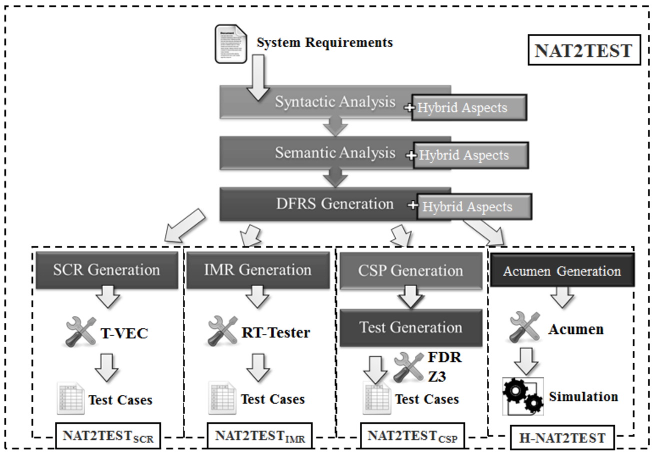

Fig. 2. Phases of the hybrid NAT2TEST strategy

ics of hybrid systems are commonly specified with the aid of mathematical expressions involving variables, functions and derivatives. Some of these features are not supported by the previous version of the SysReq-CNL. In the new version (h-SysReq-CNL), it is now possible to embed mathematical expressions in the requirements. For instance, consider the boost converter example; the following requirement describes when the system shall change its operating mode (from 1 to 2), since the switch (S) is closed ($S == 1$) and the electric charge (q) is greater than 0.

- When the system mode is 1, and $S == 1$ && $q >= 0$, the BC control system shall assign 2 to the system mode.

In Table I, one can see a fragment of the h-SysReq-CNL grammar. A *VariableState* can also be an Expression in addition to its other original forms, i.e., noun phrases, adverbs, adjectives, and numbers. The grammar rules ensure operator precedence, thus an *Expression* generates a list of *AndExpression* separated by the *OR* operator, and so on. In addition to the use of logical and arithmetic operators, the grammar allows for values, variables, expressions within parentheses and function applications. The grammar considers five built-in functions (*FunctionID*): *sin*, *cos*, *exp*, *log* and *sqrt*, besides user-defined ones. Therefore, we also allow for declaring functions (*FunctionDeclaration*), with or without parameters, which can be referred to within a requirement.

A function declaration begins with an identifier, followed by an optional list of the parameters, followed by its body. The body can be a simple expression, or a ternary expression (*TernaryExpression*), or can even be defined using pattern matching (*PatternMatching*). These last two constructs enhance the flexibility and the expressiveness of the grammar.

The ternary conditional operator is similar to an if-then-else structure: *test* ***?*** *expression1* ***:*** *expression2*, where *test* is a boolean expression, *expression1* is an expression evaluated if *test* is true, whereas *expression2* is evaluated otherwise. *PatternMatching* allows the definition of functions using pattern matching, a form widely used in the writing of mathematical equations. The structure is as follows, *(test* ***DO*** *expression1)+*, where *test* is a boolean expression and *expression1* is an expression evaluated if *test* is true.

TABLE I
FRAGMENT OF THE H-SYSREQ-CNL GRAMMAR

	...	
VariableState	→	... |Expression;
Expression	→	AndExpression (OR AndExpression)*;
	...	
Function	→	SIN LP Expression RP
		| COS LP Expression RP
	...	
		| Noun (LP ArgumentList RP)?;
ArgumentList	→	Expression (COMMA Expression)*;
	...	
FunctionDeclaration	→	Noun LP ParameterList? RP EQUALSSIGN FunctionBody;
ParameterList	→	Noun (COMMA Noun)*;
FunctionBody	→	TernaryExpression
		| PatternMatching
		| Expression;
TernaryExpression	→	Expression IN TernaryDefinition COLON TernaryDefinition;
TernaryDefinition	→	Expression
		| TernaryExpession;
PatternMatching	→	(Expression DO Expression)+;

Considering the running example, the behaviour of q in mode 1, when the switch is open and the magnetic flux in the inductor is positive, can be described by the function: $q\_mode1(Phi, L, R, C, q) = (Phi/L) - (1/R * C) * q$.

The following requirement illustrates how functions and derivates can be used to describe the system behaviour.

- When the system mode is 1, the BC control system shall: set the q first derivative to q_mode1(Phi,L,R,C,q), set the Phi first derivative to Phi_mode1(C,q,E).

The next step of the h-NAT2TEST strategy provides informal semantics for the requirements.

B. Semantic Analysis

The semantic analysis phase receives as input the generated syntax trees (obtained based on the h-SysReq-CNL grammar), and delivers a requirement semantic representation. Therefore, this phase consists of relating syntactic structures of grammar elements with semantic roles according to the case grammar theory.

In this work, we consider the same TRs adopted originally by the NAT2TEST strategy; for the verbs used in action statements: action (ACT) – the action performed if the requirement conditions are satisfied; agent (AGT) – entity who performs the action; patient (PAT) – entity who is affected by the action; TOV – the patient value after action completion. Similarly, other five roles are defined for the verbs used in conditions: condition action (CAC), condition patient (CPT), condition from value (CFV), condition to value (CTV), and condition modifier (CMD).

The mapping between words and roles is performed considering inference rules (see [12]). Since these rules are

TABLE II

EXAMPLE OF A REQUIREMENT FRAME

Condition #1 - Main Verb (CAC): is			
CPT:	the system mode	CFV:	-
CMD:	-	CTV:	1
Action #1 - Main Verb (ACT): set			
AGT:	the BC control system	TOV:	q_mode1(Phi,L,R,C,q)
PAT:	the q first derivative		
Action #2 - Main Verb (ACT): set			
AGT:	the BC control system	TOV:	Phi_mode1(C,q,E)
PAT:	the Phi first derivative		

syntax-oriented, and part of the structure of the SysReq-CNL was changed, in this work, we needed to adapt these rules considering the structure of the h-SysReq-CNL. To give a concrete example of the semantic analysis, consider the requirement shown at the end of Section III-A; the corresponding requirement frame is given in Table II.

After inferring the requirement frames of all requirements, formal semantics is given by deriving a formal model in the form of a hybrid data-flow reactive system (h-DFRS).

C. h-DFRS Generation

A data-flow reactive system (DFRS) is a symbolic, timed and state-rich automata-based notation for representing natural-language requirements. Formally, it is a 6-tuple $(I, O, T, gcvar, s_0, F)$. Inputs *(I)* and outputs *(O)* are system variables, timers *(T)* are a special kind of variable whose values are non-negative numbers representing a discrete or a dense (continuous) time. The system global clock *(gcvar)* has the same type as the timers. The initial state is *s0*, defining the initial values of variables and timers, and *F* is a set of functions describing the system behaviour. Briefly, *F* describes the expected system reaction given the current system state. For a comprehensive and formal characterisation of DFRS models, we refer to [10].

In the h-NAT2TEST strategy, the DFRS definition is modified to incorporate the user-defined functions. As a result, the h-DFRS is now formalised as a 7-tuple: $(I, O, T, gcvar, s0, F, UF)$, where *UF* denotes the user-defined functions, which can be referred to by definitions in F. The generation of h-DFRSs from requirements frames follows the same ideas of deriving DFRS models, which is fully described in [10].

D. Acumen Generation

After generating an h-DFRS, which formally represents the system requirements, we translate this model to an Acumen model in order to simulate it. The h-DFRS variables are translated to variables of a hybrid automaton, and the formal characterisation of the system behaviour (*F*), along with the user-defined functions (*UF*), are used to represent the edges of the hybrid automaton and the continuous dynamics associated with the automaton locations.

The translation from h-DFRSs to Acumen models is implemented with the aid of the *Visitor* design pattern. An h-DFRS model is represented as an XML file, and the visitor pattern visits all nodes of this representation executing the appropriate translation for each node based on the type of node, its parent and the data that it holds. In what follows, we describe the main stages of the translation.

Generating the header The initial settings of the Acumen specification is generated. Among them, we can highlight the preamble determining the simulation method (EulerForward by default, a standard technique used for discretised approximation of solutions to differential equations) and the time step (it is a pre-established value), which is the fixed and discrete increment of time that is considered when evaluating the differential or difference equations.

Generating variables Model variables are declared to represent the system inputs, outputs, and timers.

Generating auxiliary definitions Some auxiliary structures are introduced, since they are not explicitly described in the model. For example, the derivative of the *global clock* variable is set to assume a steady growth behaviour.

Generating the main model Here we use if-statements, whose conditions are composed by discrete and timed guards. The body of each if-statement comprises continuous and discrete assignments according to the statements defined in the h-DFRS (*F* and *UF*).

To illustrate the last stage, considering the DC-DC boost converter, we show part of the Acumen code generated for the requirements previously mentioned (reproduced below).

- When the system mode is 1, and $S == 1$ && $q >= 0$, the BC control system shall assign 2 to the system mode.

```
if (the_system_mode==1 && (S==1 && q>=0))
    then the_system_mode += 2
```

In the Acumen language, += is used to perform discrete assignments (as opposed to continuous ones, denoted by =).

- When the system mode is 1, the BC control system shall: set the q first derivative to q_mode1(Phi,L,R,C,q), set the Phi first derivative to Phi_mode1(C,q,E).

```
if (the_system_mode==1) then
    q' = (Phi/L)-(1/R*C)*q,
    Phi' = (-1/C)*q + E
```

We note that, in the last example, the function calls were inlined, and that q' and Phi' denote the first derivative of q and Φ, respectively. After visiting all elements of the h-DFRS, a complete Acumen specification of the system behaviour is generated automatically.

IV. CASE STUDY

The h-NAT2TEST strategy is supported by an extension of the original NAT2TEST tool[1]. We illustrate the applicability of our tool by applying it to the requirements of the DC-DC boost converter (BC). Considering that a typical BC has four operating modes according to the current position of the switch and whether the diode *D* is conducting/blocking energy, we defined auxiliary functions for each mode. These

[1]http://www.cin.ufpe.br/~ghpc/

functions are referred to in the description of the continuous behaviour of the system (i.e., how q and Φ—the electric charge of the capacitor and the magnetic flux of the inductor—evolve continuously). Therefore, eight functions were created in total. In what follows, one can see the functions related to q and Φ in the first mode.

FUN-001 q_mode1(Phi,L,R,C,q) = (Phi/L) - (1/R*C)*q.
FUN-002 Phi_mode1(C,q,E) = (-1/C)*q + E.

Taking into account these functions, four requirements were written to describe the continuous evolution of the system (the continuous assignments performed within each operating mode). To conclude our specification, twelve requirements were written to describe the system's discrete behaviour: the discrete changes between the operating modes. Therefore, sixteen requirements were written in total: four describing the continuous dynamics and twelve concerning the discrete transitions. Two of these sixteen requirements are presented in the end of Section III.

To analyse whether the specification describes a DC-DC Boost Converter that is capable of delivering a boosted and stable output voltage, we employ the h-NAT2TEST to simulate the natural-language requirements. After checking whether each system requirement is valid with respect to the h-SysReq-CNL grammar, thematic roles are inferred from the obtained syntax trees. Afterwards, an h-DFRS is derived from the previously obtained requirements frames. Finally, an Acumen specification is generated and we use its environment to perform simulations.

Considering our first specification, as it can be seen from Fig. 3, the system global clock grows steadily, and the switch changes discretely and periodically between "on" and "off" positions. However, although such a frequency of open/close produces a boosted output voltage (see q), it is not a steady one. In practice, this high oscillation may not be acceptable.

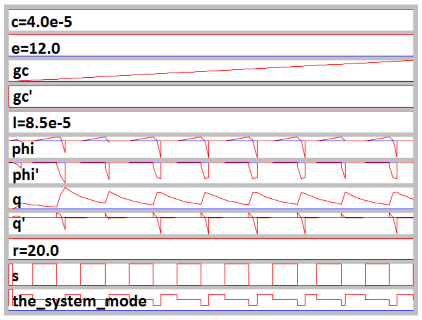

Fig. 3. Specification #1 – simulation of the Acumen model

Inferring such a behaviour (high oscillation of the output voltage) purely from reading the natural-language requirements is not an easy task. In order to have a more steady output, the specification was revisited. Fig. 4 shows the simulation obtained from this second specification. As one can see, a higher frequency of open/close can make the output voltage more stable. This stabilisation is an expected behaviour in normal conditions. One may, of course, use more sophisticated controllers to achieve stability.

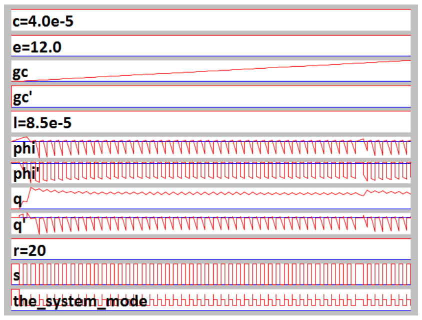

Fig. 4. Specification #2 – simulation of the Acumen model

This example illustrates the h-NAT2TEST applicability, where natural-language specifications are turned into executable models and, based on simulations, one can validate whether the specification describes the intended behaviour.

V. Related Work

There are many tools available for modelling, simulating and verifying properties of hybrid systems. In what follows, we review some of them.

Simulink [13] is an interactive environment for modelling, simulating, and analysing multidomain dynamic systems. It supports linear and nonlinear systems, modelled in continuous time, sampled time, or a hybrid of the two. It is widely used in many industrial domains for modelling and analysing various types of CPSs.

20-sim [3] is a modelling and simulation program for mechatronic systems. The model is built graphically by drawing an engineering scheme. It is possible to create models using equations, block diagrams, physical components and bond graphs.

Modelica [4] is an object-oriented language for modelling heterogeneous physical systems for the purpose of simulation. For modelling the physical phenomena, it uses general equations instead of assignment statements.

SpaceEx [14] is a verification platform for hybrid systems. It uses components built as hybrid automata as input.

S-TaLiRo [15] is a tool to verify and test CPSs. It is a modular software tool built on the Matlab platform. S-TaLiRo can analyse hybrid automata, arbitrary Simulink models, hardware-in-the-loop and processor-in-the-loop.

In addition to these established tools, in [7] the authors present a tool prototype for model-based testing of cyber-physical systems. It is implemented in Matlab and comprises the three stages of model-based testing, namely, test case generation, test case execution, and conformance testing. The specification and implementation models can be provided as either Matlab or Acumen models.

Despite the existence of numerous tools that work with hybrid systems, to the best of our knowledge, all available tools require concrete representations of hybrid systems such as hybrid automata or block diagrams as their input models. Therefore, in the preliminary stages of system development, when typically these models are not yet available, but rather the system specification in the form of natural-language requirements might be the only documentation available, these tools cannot be readily used.

In order to bridge this gap, we proposed in this paper the h-NAT2TEST strategy. It is important to emphasise that this strategy does not replace the tools previously mentioned; the aforementioned tools are still essential when analysing later system designs, but our tool complements them by allowing for simulation using early natural-language descriptions.

VI. Conclusion

The major motivation for this work is to provide means for simulating natural-language requirements of hybrid systems, in order to enable early validation of these requirements. In [6], a strategy (NAT2TEST) is proposed for analysing and testing reactive systems from natural-language requirements. Here, we extend this work to the domain of hybrid systems. The strategy proposed here (h-NAT2TEST) extends the first three stages of the original one: syntactic and semantic analysis, and generation of data-flow reactive models.

In the syntactic analysis, we propose a conservative extension of the controlled natural language SysReq-CNL (h-SysReq-CNL), adding to it the manipulation of expressions, besides declaring and referring to user-defined functions. In the semantic analysis, the inference rules of thematic roles were updated considering the new elements now supported by the h-SysReq-CNL grammar. Moreover, we now consider a hybrid data-flow reactive system (h-DFRS), suitable for formally representing hybrid systems.

Finally, a translation from h-DFRSs to Acumen models was implemented, allowing the user to simulate natural-language requirements of hybrid systems via the Acumen tool, without the knowledge of the intermediate notations. We illustrated this strategy considering a DC-DC boost converter, whose application was successful and led us to revise our first specification in order to achieve a description of how to deliver a boosted and steady output voltage.

As future work, we plan to address the following topics.

- *Represent environment properties as natural-language*. Currently, assumptions of the system environment (e.g., how the system inputs might evolve over time) are not described in natural-language and, if required, need to be manually specified in the Acumen model. We plan to extend our CNL to be capable of also describing such properties, besides adapting the strategy in order to generate Acumen models with such information automatically.
- *Expand the proposed approach to performing conformance testing*. Conformance testing is a formal notion of model-based testing. Conformance is defined as a mathematical relation between the expected (specification) and observed (implementation) behaviour. In this direction, we plan to integrate our work with the tool described in [7], which also deals with Acumen models.
- *Validate the strategy in an industrial context*. In order to demonstrate the technique effectiveness, perform more empirical analyses concerning the application of the h-NAT2TEST strategy to real-world large-scale examples of cyber-physical systems.

Acknowledgements

The work of Bruno Oliveira, Gustavo Carvalho and Augusto Sampaio was partially supported by the CIn-UFPE/Motorola cooperation project, as well as CNPq grants 303022/2012-4 and 132332/2015-9.

The work of Mohammad Reza Mousavi has been supported by (Vetenskapsrådet) award number: 621-2014-5057 (Effective Model-Based Testing of Concurrent Systems), by the Swedish Knowledge Foundation (Stiftelsen for Kunskaps- och Kompetensutveckling) in the context of the AUTO-CAAS HÖG project (number: 20140312), and by the ELLIIT Strategic Research Environment.

References

[1] Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. An approach to the description and analysis of hybrid systems. In *Hybrid Systems*, pages 149–178. Springer, 1993.

[2] Changho Sung and Tag Gon Kim. Framework for simulation of hybrid systems: interoperation of discrete event and continuous simulators using HLA/RTI. In *Proceedings of the 2011 IEEE Workshop on Principles of Advanced and Distributed Simulation*, pages 1–8. IEEE Computer Society, 2011.

[3] Jan F Broenink. 20-sim software for hierarchical bond-graph/block-diagram models. *Simulation Practice and Theory*, 7(5):481–492, 1999.

[4] Peter Fritzson and Vadim Engelson. Modelica – a unified object-oriented language for system modeling and simulation. In *European Conference on Object-Oriented Programming*, pages 67–90. Springer, 1998.

[5] Walid Taha et al. Acumen: An open-source testbed for cyber-physical systems research. In *International Internet of Things Summit: Internet of Things. IoT Infrastructures*, pages 118–130. Springer International Publishing, 2016.

[6] Gustavo Carvalho, Flávia Barros, Ana Carvalho, Ana Cavalcanti, Alexandre Mota, and Augusto Sampaio. NAT2TEST Tool: from Natural Language Requirements to Test Cases based on CSP. In *International Conference on Software Engineering and Formal Methods*. Springer International Publishing, 2015.

[7] Arend Aerts, Mohammad Reza Mousavi, and Michel Reniers. A tool prototype for model-based testing of cyber-physical systems. In *Proceedings of the 12th International Colloquium on Theoretical Aspects of Computing - ICTAC 2015 - Volume 9399*, pages 563–572, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

[8] Thomas A Henzinger. The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*, pages 265–292. Springer, 2000.

[9] Charles J. Fillmore. The Case for Case. In Bach and Harms, editors, *Proceedings of Universals in Linguistic Theory*, pages 1–88, USA, 1968. New York: Holt, Rinehart, and Winston.

[10] Gustavo Carvalho, Ana Cavalcanti, and Augusto Sampaio. Modelling timed reactive systems from natural-language requirements. *Formal Aspects of Computing*, 28(5):725–765, 2016.

[11] Bruno Oliveira. Simulation of hybrid systems from natural language requirements. Master's thesis, Universidade Federal de Pernambuco, 2016.

[12] Gustavo Carvalho, Diogo Falcão, Flávia Barros, Augusto Sampaio, Alexandre Mota, Leonardo Motta, and Mark Blackburn. NAT2TEST$_{SCR}$: Test case generation from natural language requirements based on SCR specifications. *Science of Computer Programming*, 95, Part 3(0):275 – 297, 2014.

[13] Mathworks. Simulink: User's guide, 1995.

[14] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*, pages 379–395. Springer, 2011.

[15] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram 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*, pages 254–257. Springer, 2011.