Qualitative simulation and validation of complex hybrid systems

Jean-Pierre GALLOIS and Jean-Yves PIERRON

CEA, LIST, Laboratoire d'Ingénierie Dirigée par les Modèles pour les Systèmes Embarqués

Point Courrier 174, Gif-sur-Yvette, 91191, France (e-mail: jean-pierre.gallois@cea.fr, jean-yves.pierron@cea.fr)

Abstract

Complex industrial systems need extensive validation and verification. Methods for this are well advanced in case of discrete systems. However, for hybrid systems that combine discrete and continuous aspects, they are not as well developed. To deal with this, qualitative simulation can be used, based on the principle of discretization by identifying domains of variation of continuous variables and tracking the evolution of these variables. A system can be discretized by representing its continuous parts, which are described by differential equations. When these are coupled with the discrete parts of the system, a fully discrete global model is obtained, on which formal techniques can be applied for the validation process. If the differential equations cannot be expressed clearly, it is necessary to establish a qualitative model describing the laws of evolution of continuous variables. We defined and tested a novel methodology that represents variations of continuous variables and the causal links between them to obtain mappings of system behaviors that are suitable for validation.

Keywords: hybrid systems, formal methods, qualitative simulation, validation

1. Introduction

Industrial systems are becoming increasingly more complex, requiring more powerful formal methods of verification and validation. Large models are usually difficult to solve analytically, so engineers use numerical simulations to study their behavior. But numerical simulation has limitations. It requires determining the values of parameters and initial conditions. In practice, these are often difficult to specify. Another difficulty is needlessly calculating the behavior of a system for parameter values that are of no practical interest in a lot of situations. More often it is of greatest interest to determine what class of behaviors is to be expected for a given constraint imposed on the initial conditions and parameters. That is why qualitative models and simulations have been used; they can cope with imprecise knowledge and with infinite numerical value domains by compacting them into abstractions, and using qualitative predictions to predict the significant qualitative classes of system behaviors. This type of qualitative reasoning is also suitable for process analyses and verifications.

In this paper we describe an approach proposed by CEA LIST, which uses formal techniques in order to automatically prove safety properties and to automatically generate a set of corresponding test scenarios. To that end, a methodology and a corresponding tool (named DIVERSITY) based on symbolic execution were developed at CEA LIST. This technique allows generating symbolic scenarios corresponding to classes of system behaviors, which are sequences of well-defined actions. Once the set of all possible symbolic scenarios is computed, it is possible to prove properties on this set and to generate concrete numerical tests from them (a single numerical test is sufficient to represent a given symbolic scenario). Symbolic execution is a proven way of overcoming both combinatorial explosion and computational redundancies. However, to analyze complex systems that include continuous components that are classically modeled by hybrid automata, it was necessary to extend the simulation method used in the DIVERSITY tool to support qualitative methods.

Qualitative reasoning has been applied in different domains as Artificial Intelligence, mathematics, economy, and much in bioinformatics this last decade [MGCL07]. Recent developments of formal tools and the increasing power of computers lead us to believe that these methods can be applied effectively to industrial-size hybrid systems.

State of the art

Hybrid systems can be described by a finite set of continuous processes and transitions, and a set of real and discrete variables on which they operate. The processes are generally defined by differential equations that give changes in the real variables. Transitions are discrete assignments and guarded with variables.

There are several classes of hybrid systems: if the differential equations, guards and assignments are linear, then the hybrid automaton is said to be polyhedral (polyhedral hybrid automaton). If the differential equations are equalities and inequalities that include constant parameters, then it is called rectangular (rectangular hybrid automaton). Finally, if the differential equations are equalities and inequalities that include the unit of time as a parameter, the system is said to be timed (timed automaton) and real variables are clocks (clocks). Timed automata are specially handled by the Uppaal [Be02] and Kronos [Bo98] tools.

For other types of hybrid systems, we can mention the following tools and methods, which are used to calculate a set of accessible states approximated by a superset of those states: Hytech [HHT97] for rectangular hybrid automata with transitions with linear guards; Checkmate [CK03] for non-linear continuous evolutions but with linear guards, evolutions approximated by a polyhedral; the Tiwari & Khanna method [TK02] for polynomial continuous evolutions, where the system is abstracted as a discrete system whose guards are polynomial. This latter approach [TK02] was chosen in our work because it can manipulate polynomial constraints that are widely used in differential equations involved in embedded systems, especially in the mechanical domain. The QEPCAD tool was used to deal efficiently with constraints including polynomial expressions [Br03].

If the system is not clearly described by differential equations, a qualitative model must be established using specifications that describe the evolution of continuous variables. Many works have been published on this topic and a good survey of them has been presented in “Modèles et raisonnements qualitatifs” [TD03], and, prior to that, in “A Survey of Techniques and Applications” [Qr95].

Technical context

A hybrid automaton [H96] is defined by a set of states and discrete and continuous variables. A transition between states has a guard that depends on the values of the variables and will assign new values to the variables corresponding to the new target state. For continuous variables, the laws of evolution can be described by more or less accurate differential equations in the context of the various possible formalisms and methods [TD03]. Continuous variables change according to these laws, representing the different states of the system. System states are timed in the sense that they have a duration. In contrast, the transition time between states of the system is assumed to be zero, as the system performs instantaneous assignments to variables.

To simulate this type of model, qualitative simulation [K86] [TK02], which is an alternative to numerical simulation is used. It is based on the principle of discretization by partitioning domains of variation of the continuous variables, based on the evolution trends of these variables (increasing, decreasing, or constant) indicated by the signs of their first derivatives (positive, negative or null). In this way, one can get a tree of abstract behaviors [RGLG03] that allows for coverage of the system states [GP14]. If these are coupled with the discrete part of the system, a fully discrete global model is obtained on which formal techniques can be applied.

If the differential equations are not available, it is necessary to establish a *qualitative model* that describes the laws of evolution of the continuous variables. One solution is to represent the velocity of variations for continuous variables and to establish causal links between them. This construction of automata modeling the evolution of continuous variables requires specific rules that we present here. These rules allow the definition of qualitative states and associated transitions where each qualitative state corresponds to a set of values for continuous variable with regards to its evolution law (corresponding to the sign of the derivative: positive, negative or null). Each transition defines a rule that allows the evolution to move from one set to another (e.g., move from a zero to a positive derivative).

2. Models with differential equations

As part of our study we retain the framework of models with differential equations of the first order (i.e., with first derivatives only), with polynomial integer coefficients, but to any degree and with any number of variables (this is the framework chosen by Tiwari [TK02] for qualitative simulation). Higher-order differential equations can be reduced to first-order differential equations by changing variables.

In summary, continuous variables change continuously with time and they can be calculated by the differential equations applied to the control states which define the laws of change. When a guard of a transition whose source state is the current state of control becomes true, this transition is fired, resulting in new values being assigned to certain variables. In the new state of control, which is the target state of the transition, the new differential equations applying to that state provide the corresponding laws for the further evolution of the continuous variables.

Qualitative simulation with differential equations

In this section, the context is assumed to be a continuous system described solely by differential equations.

Qualitative simulation, whose core principles were well defined in the 80s and which was refined in the 2000s, is based on the principle of discretization by partitioning areas of variation of the continuous variables of the system, based on their evolution trends (increasing, decreasing or constant) based on the signs of their first derivatives (positive, negative or zero). When all the discrete states corresponding to this qualitative partitioning are created, we can apply a simple algorithm based on the possible evolution of each variable. That is, a derivative may change its sign only when vanishing, because the process is continuous (for example a positive derivative must first be zero before becoming negative). This limits possible evolutions and thus reduces the size of the corresponding graph. Finally, the differential equation system allows generation of a transition system whose states are based on partitioning continuous variables into change in areas as described above and whose transitions are possible evolutions of these states. It has been shown that the result of such a qualitative simulation is an abstraction of the original differential equation system.

Symbolic execution

In the following, we describe the classical technique of symbolic execution used in the DIVERSITY tool as it is a based on the method presented in this paper, with an adaptation to hybrid automata context.

Starting from a transition system whose transitions are labeled by conditions and assignments to variables, it is possible to produce its corresponding symbolic execution tree, in which variables are described by expressions with parameters.

Given the current state S of the transition system, and a condition which is associated with the current state, the path condition PC, a symbolic execution constructs the tree of sequences of symbolic states, based on the following rules:

The root of the tree is the initial state (the current state at the beginning of the process);

For each possible successor of the current state S, a path can be constructed, if and only if the condition C on the parameters that makes a state S’ a successor of the current state can be true; in that case, an edge is constructed from S to S’ and the new symbolic values of variables are evaluated, applying the assignments, their symbolic values having been updated by substitution and simplified by a simplification procedure (for instance, if x + y is assigned to x, with a and b as initial values of x and y, then the final value of x will be a + b);

S’ is associated with a new constraint PC’, which is the conjunction of PC and C (if the PC is evaluated to false by a constraint solver then the path is cut);

The process is repeated with the successors of S';

If a new symbolic state has already been reached along the same path (or in a previously calculated path if a more compact tree is required), then the execution of this path stops: this can be determining by comparing the appropriate sets of variables, by the criteria of inclusion (or equality if more precise calculus is required), for instance if the PC is $x>0$ in the previous state, and PC' is $x > 1$ in the final state, then the set of possible values of x in the final state is included in the previous state and the path can be cut;

The symbolic execution is over when all the possible paths have been assessed.

The discretization process

The discretization process requires creating a state machine for each variable representing an abstraction of its evolution. A variable x can be increasing, decreasing or constant, which is equivalent to the differential equations $dx/dt > 0$, $dx/dt < 0$, $dx/dt = 0$. This will be represented by a state machine with 3 states and the possible transitions between them corresponding to possible evolutions of the system.

Each transition is defined by an initial state, a guard, a final state, and a constraint expressed by a formula that must be evaluated a posteriori and whose evaluation will be based on each variable value after having been updated by its own state machine. The update of the variable depends on its direction of change (e.g., the new value of x > old value of x if $dx/dt > 0$ ...). As the constraints represent the differential equations according to the direction of evolution of the synchronized variables, the conjunction of these constraints represents the overall constraint that must be satisfied if the overall step of simulation is possible. To evaluate this overall formula the tool QEPCAD was used, which handles polynomial constraints. If the formula evaluates to false, the corresponding branch in the execution tree is cut. This process is repeated until the entire execution tree has been computed. As control states correspond to the direction of variation of each variable, we obtain for 3 states for 1 variable, and 3n states for n variables. As the set of states is finite, the process will necessarily terminate. This number of states (3n) gives a good idea of the potential complexity of the resulting models. It is clear that, in the worst case, the number of paths may be excessive. However, we assume that, if the method is applied to models of real-world systems, the number of classes of behaviors represented in this way will generally not be too large.

Complete system

In the case of a "traditional" hybrid system, the differential equation system which describes the continuous part of the system is combined with a discrete system in which there are discrete variables and transitions that affect these variables. For the qualitative simulation the discrete part of the system is unaffected by the discretization method described above. Control statements are for discrete variables "simple" states, while for continuous variables they are continuous process. The latter can be discretized with the qualitative simulation mechanism described above. Consequently, the entire resulting system is discrete, which means that it can be treated using traditional mechanisms (verification, test generation) that are applicable to discrete systems.

Nevertheless, there is a case which can be problematic: it occurs when the discrete part of the system contains variables whose domains of definition are infinite. Indeed, the number of states of the system in that case is potentially infinite, limiting the ability to fully analyze the system. In practice, if one is faced with such systems, several techniques are possible. We have chosen symbolic execution, which provides a representation that is either equivalent to the real system or a valid abstraction of it. It has been shown that when symbolic execution is applied, if the cut-off criterion is the equality of domains, then the execution tree is bi-similar to the digital implementation of the numerical tree. On the other hand, if the cut-off criterion is the inclusion of domains, then the symbolic execution tree is an abstraction of the numerical implementation of the system tree.

Working with Booleans or integers in the scope of Presburger arithmetic ensures the decidability of the process of the equality or inclusion. In this context, applying the symbolic execution to the discrete part of the system produces at least an abstraction of the execution tree of the system. As qualitative simulation also produces an abstraction of differential equations that it processes, the product of qualitative simulation and symbolic execution gives an abstraction of the execution tree of the system.

If we place ourselves in a less constrained framework, for example in the context of handling complex calculations of real numbers, we cannot apply the preceding cut-off criteria for the calculation of the symbolic execution tree (equal criterion or inclusion). In that case, the cut-off criterion chosen is generally a criterion of depth in the execution tree. In this case, we know that to this depth the symbolic execution tree is bi-similar to the numerical execution tree. It is thus necessarily an abstraction, and, therefore, for a chosen depth, we have as in the previous case of Presburger arithmetic, the ability to produce an abstraction of the numerical tree of the system.

As a result, we have an original method and an associated tool – based on symbolic execution for discrete parts of models and qualitative simulation for continuous parts – which can provide a usable abstraction of models of complex systems in every case. This approach reduces path redundancies to a minimum and, thereby, provides efficient support for analysis and verification of system properties.

Implementation

Qualitative simulation produces a discrete graph that is an abstraction of the system of the represented differential equations. The discrete graph is implemented by a transition system expressed in XLIA, which is the internal language of DIVERSITY.

Similarly, the discrete parts of the system can be translated from a model expressed in an industry-standard language like UML, SDL, or Matlab/Simulink, into XLIA, to be processed by the DIVERSITY tool.

Therefore, it is planned to jointly run the two systems (discretized differential equations and discrete system) translated into XLIA in a model by creating a model that includes these two XLIA models along with the connections between them. In the preceding discussion, it was assumed that the system has just a single differential part; but one can generalize this reasoning to a system with multiple differential parts, each of which can be discretized using the same process and then translated to XLIA.

An example: The Brusselator

The Brusselator is an example of an oscillating autocatalytic chemical reaction [AH03]. Its mechanism is given by the four following chemical equations:

1. $A \to X$
2. $2X + Y \to 3X$
3. $B + X \to Y + C$
4. $X \to D$

The dynamics of this system are described using differential equations and are typically resolved by a Jacobian.

Our goal is to provide a discrete description of the dynamics of the Brusselator. For this we divide the plane (X, Y) in areas according to increase/decrease X and Y, wherein these domains are synthesized by abstract states.

We thus obtain the graph of abstract states of X and Y, and therefore the behavior of the Brusselator system.

Equations

$Vx = 1 - (b + 1) X + a X^2 Y$ (the growth rate of the concentration of X)

$Vy = b X - X^2 Y$ (the growth rate of the concentration of Y)

Note 1: variables X, Y, and parameters a and b are positive.

Note 2: If the speed Vx is positive [resp. negative], the concentration of X increases [resp. decreases].

Abstract space

Division of space:

- 3 possible states for Vx (Vx = 0, Vx> 0 or Vx <0);

- 3 possible states for Vy (Vy = 0, Vy > 0 or Vy <0).

This makes a total of nine states (3x3):

State S1: Vx < 0 and Vy > 0

State S2: Vx = 0 and Vy > 0

State S3: Vx > 0 and Vy > 0

State S4: Vx > 0 and Vy = 0

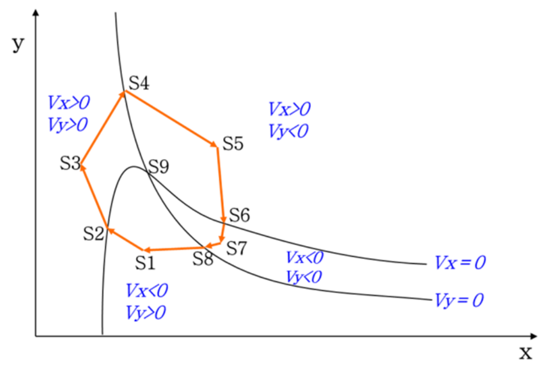
State S5: Vx > 0 and Vy < 0

State S6: Vx = 0 and Vy < 0

State S7: Vx < 0 and Vy < 0

State S8: Vx < 0 and Vy = 0

State S9: Vx = 0 and Vy = 0



The diagram represents the qualitative simulation of the Brusselator.

Dynamic analysis

Calculating all the possibilities of movement is equivalent to firing all transitions from each abstract state of the abstract space listed above. The simulation then yields the tree of all possible paths, which captures all possible dynamic behaviors.

Qualitative simulation shows strictly and explicitly the loop operation, which is characteristic of the Brusselator, without having to unfold the standard calculations of numerical analysis (by the Jacobian). On the other hand, the resulting abstract model provides an excellent support for the proof (e.g., it can show a loop clearly, or evaluate the intersection of the set of qualitative states with a forbidden region), especially for temporal logic properties. Finally, the abstract model can also help in the definition of test sequence (in this case, it demonstrates a characteristic sequence of the main loop by an experiment).

3. Models without differential equations

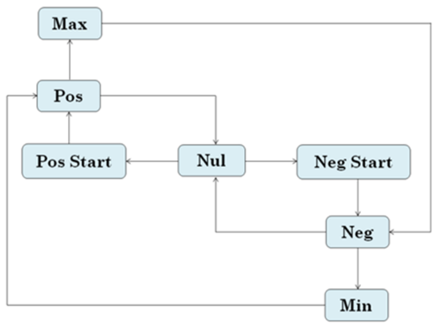
The method can be applied to a system including the physical data subsystem that is continuous. For this type of model the evolution of continuous variables laws can be described by differential equations or by abstract equivalents based on the tables of variation. In this section, we focus on the second case. The objective is to realize a hybrid automata representation of the system, and to apply to this model the technique of qualitative simulation that provides classes of behavior. The value of having such classes is to gain generality when validating the simulated model. Thus, if a loop is detected, it can be deduced that this loop exists for all the corresponding numerical paths. A major application of the qualitative method is getting a "mapping" of the behavior of the system. This mapping is intended to serve as a support for other purposes, such as co-simulation, HMI validation, etc.

Modeling the hybrid system

The inputs to the tool are specifications (which may be textual) and known data of the system, including engineering experiences. The model can then be encoded directly into XLIA, the internal language of DIVERSITY, in the form of concurrent state machines. It may be encoded using an industry-standard such as

SysML (an objective for our future work), but it is necessary to translate this to XLIA to use the DIVERSITY tool.

When the differential equations of a system are not available to describe the behaviors of the continuous variables, their laws must be expressed by hybrid automata, which must conform to specific methodological construction rules. These rules are summed up by the following diagram, which represents the possible sequences of qualitative states.



Indeed, they may be to the Nul state, their values are constant. They can begin to grow, which corresponds to the Pos Start state, or begin to decline, which corresponds to the Neg Start state. They may continue to decline, which corresponds to the Neg state, or continue to grow, that is to say the Pos state, eventually reaching either the Max or the Min state.

From the XLIA model DIVERSITY generates a tree of behaviors that can be represented in a graphical format for easier comprehension. For example, system cycles can be highlighted in this way.

Thus execution traces resulting from DIVERSITY simulations of the model can be analyzed and compared with potential scenarios configured by the operator in the simulator. In addition, one can compare the execution traces produced by the numerical simulator to what is expected by the qualitative model, to establish compliance of the numerical simulation to the qualitative simulation. DIVERSITY includes a feature that can automatically generate a compliance verdict.

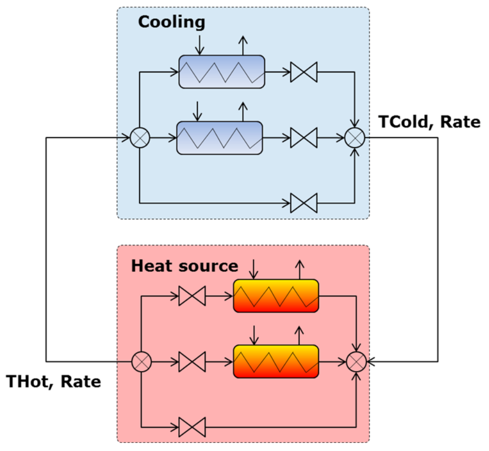
Outputs

DIVERSITY calculates a symbolic execution tree in which each execution path corresponds to a potential system behavior. If test is provided for each such path, a satisfactory coverage of behaviors can result.

The classical system properties that can be identified with this technique include detecting cycles (e.g., oscillations) and reachability (or not) of specific states (critical, non-critical, etc.).

Case study

This technique was applied to the SRI temperature control model (Intermediate Cooling System [NDB14]), which was coded directly into the internal language of DIVERSITY. The SRI is a basic system found in French nuclear power plants, which is used to provide cooling of auxiliary secondary circuits connected to the turbine process. This is a device for cooling several hot sources through an interface with a cold source, controlled by exchangers. It is of reasonable size and is sufficiently independent without being too simple, permitting it to be used in “Cluster Connexion” (a French BGLE2 project, COntrôle commande Nucléaire Numérique pour l’EXport et la rénovatION)) with available project resources. It contains logical and analog parts, and is, therefore, a good candidate for our study, which was coded directly into the internal language of DIVERSITY using the modeling methodology described above.



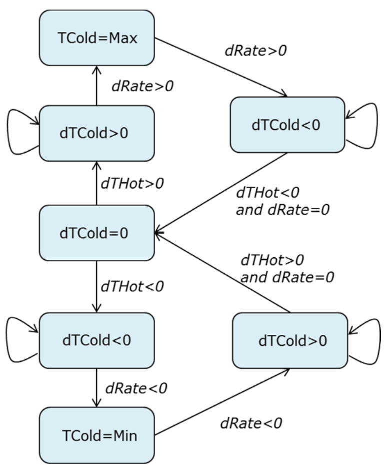
Three state variables in the SRI model were the primary subject of the simulation process study: the hot temperature of the circuit (input to exchangers), the cold temperature (output from exchangers) and the flow in the exchangers (given by the rate of the flow within an exchanger). All other parameters were assumed to be constant.

From the qualitative SRI model DIVERSITY generated a behaviors tree that could be represented in a graphical format for readability. This set of paths can highlight any potential oscillation in the control cycle, which can occur when the set temperature is exceeded, giving rise to several control cycles before stabilizing.

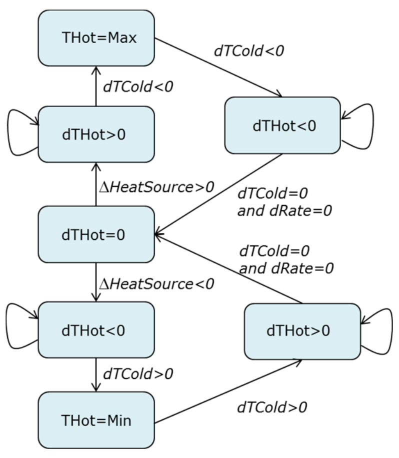
Another useful input to the DIVERSITY tool is a set of traces based on numerical simulation scenarios of the system. These traces capture observable variables in the numerical simulations. They can be used to demonstrate compliance of the numerical model with the XLIA model. Namely, the execution traces of the simulator can be compared to those of the qualitative model so that compliance of the numerical simulation with the qualitative simulation modulo these traces could be established. This was done by considering changes in temperature measurements in the circuit (by observing the oscillatory phenomenon of the regulation).

The following are graphical representations of the three main automata of the model that was realized in DIVERSITY and the graphical representation of the results of the qualitative simulation calculated by the tool:

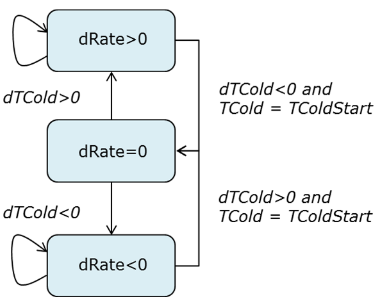
Cold Temperature: TCold.
Variation of TCold: dTCold



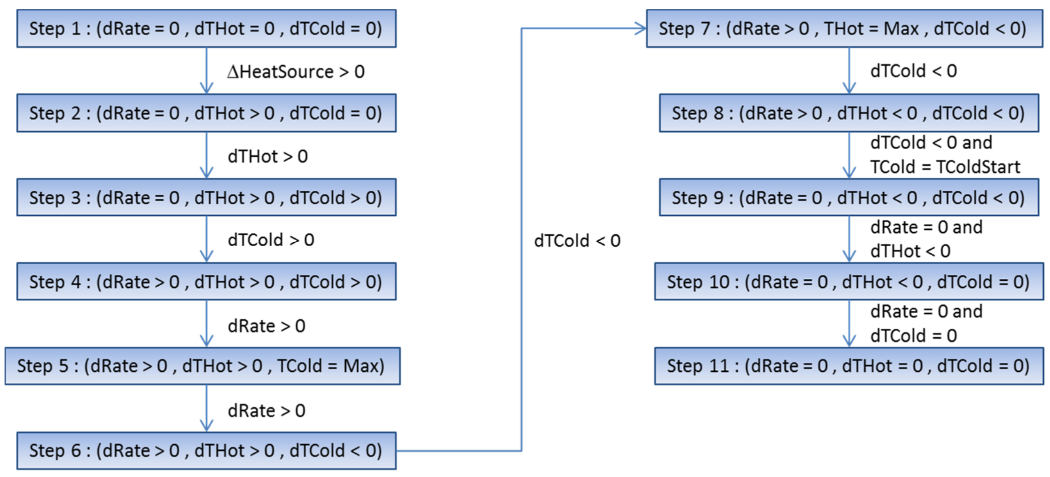
Hot Temperature: THot.
Variation of THot: dTHot



Exchange rate: Rate.
Variation of Rate: dRate



Note: ΔHeatSource>0 (resp. < 0) means an increasing (resp. decreasing) of heat source.
Thus we finally obtain a set of paths containing all potential behaviors. The following picture describes one of them.


Step 1 : (dRate = 0 , dTHot = 0 , dTCold = 0)
ΔHeatSource > 0
Step 2 : (dRate = 0 , dTHot > 0 , dTCold = 0)
dTHot > 0
Step 3 : (dRate = 0 , dTHot > 0 , dTCold > 0)
dTCold > 0
Step 4 : (dRate > 0 , dTHot > 0 , dTCold > 0)
dRate > 0
Step 5 : (dRate > 0 , dTHot > 0 , TCold = Max)
dRate > 0
Step 6 : (dRate > 0 , dTHot > 0 , dTCold < 0)
dTCold < 0
Step 7 : (dRate > 0 , THot = Max , dTCold < 0)
dTCold < 0
Step 8 : (dRate > 0 , dTHot < 0 , dTCold < 0)
dTCold < 0 and
TCold = TColdStart
Step 9 : (dRate = 0 , dTHot < 0 , dTCold < 0)
dRate = 0 and
dTHot < 0
Step 10 : (dRate = 0 , dTHot < 0 , dTCold = 0)
dRate = 0 and
dTCold = 0
Step 11 : (dRate = 0 , dTHot = 0 , dTCold = 0)

For this global case study we obtained a behaviors tree containing 49 states and 18 paths. Each path is a behavior cycle of the system. The longest path contains 19 states.

The complexity of the approach without using differential equation is similar to the complexity encountered with differential equations, in terms of numbers of states and potential paths of the produced tree. Based on that so we anticipate that, for reasonable case studies we will not have to model systems with very large numbers of distinct classes of paths, and that the method will remain practical. This will have to be evaluated more in the future with larger experiments.

4. Conclusion

The technique of qualitative simulation with differential equations has been evaluated using the CEA LIST DIVERSITY tool on several different models, including a model of chemical equations (the Brusselator), which is completely nonlinear. The main result of this was that a mapping of the behaviors of such systems can be calculated very quickly and presented in an abstract form, using qualitative states and transitions. For instance, in the chemical model example we obtained a single path which synthesized many concrete numerical paths.

Applying the method to the case when differential equations cannot be expressed clearly was also evaluated. A model of the cooling system in the French BGLE2 project "Cluster CONNEXION" was used. It provided interesting results, allowing the comparison of qualitative paths against the numerical paths obtained by numerical simulation, which are difficult to classify for validation activities. Since the qualitative paths are an abstraction of the numerical ones, they are more usable for formal activities and provide a more human-friendly and more suitable representations of behaviors (e.g., detecting oscillations or proving essential properties of the system).

Prospect

A tree produced by DIVERSITY can be used in co-simulation to validate other systems that interact with the hybrid system represented in this way. Indeed, co-simulation generally involves numerical simulators which often involve long computation times and which are necessarily configured only for specific scenarios, thus reducing the scope of exploration. In contrast, qualitative simulation provides a good abstraction of all system behaviors, which requires less computation time and thereby enables more exhaustive exploration.

We plan to construct a component FMU (Functional Mockup Unit) from the execution tree generated by DIVERSITY, which can play the role of observer and actuator, and which can be interfaced with a simulated system. This component can then be used to automatically drive a pre-computed execution scenario replacing the need for a human operator of the simulator.

Bibliography

[AH03] Ault, Shaun; Holmgreen, Erik. "Dynamics of the Brusselator" Academia.edu, 16 March 2003. 11/27/10 http://fordham.academia.edu/ShaunAult/Papers/83373/Dynamics_of_the_Brusselator

[Be02] Bengtsson, J., Griffioen, W. O. D., Kristoffersen, K. J., Larsen, K. G., Larsson, F., Pettersson, P., and Yi, W. (2002). Automated verification of an audio-control protocol using UPPAAL. Journal of Logic and Algebraic Programming 52-3, 163-181.

[Bo98] Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., and Yovine, S. (1998). Kronos: A model-checking tool for real-time systems. In Computer Aided Verification, LNCS 1427, 546-550, Springer.

[Br03] C.W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. SIGSA Bulletin, 37 (2003), pp. 97–108

[CK03] Chutinan, A., and Krogh, B. H. (2003). Computational techniques for hybrid system verification. IEEE Transactions on Automatic Control 48, 64-75.

[Qr95] Qualitative reasoning: A survey of techniques and applications, P. Bourseau, K. Bousson, P. Dague, J.L. Dormoy, J.M. Evrard, F. Guerrin, L. Leyval, O. Lhomme, B. Lucas, A. Missier, J. Montmain, N. Piera, N. Rakoto Ravalontsalama, J.P. Steyer, M. Tomasena, L. Trave-Massuyes, M.R. Vescovi, S. Xanthakis, A. Yannou, AI Communications. Special Issue MQ&D: Qualitative Reasoning, Vol.8, N°3/4, pp.119-192, Sept-Dec 1995

[GP14] Jean-Pierre Gallois et Jean-Yves Pierron, INTERVAL, instanciation d'une plate-forme de validation pour les spécifications industrielles dans le cadre du projet CONNEXION, Génie Logiciel Hors-série « L'initiative Connexion : IDN et Contrôle-Commande »: 32-38, 2014.

[H96] T. Henzinger, The Theory of Hybrid Automata, Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 96), pp. 278-292.

[HHT97] Henzinger, T. A., Ho, P. H., and Toi, H. W. (1997). HYTECH: A model checker for hybrid systems. STTT 1(1/2), 110-122.

[K86] KUIPERS B.J. (1986). Qualitative simulation. Artificial Intelligence vol 29 n3 239-388.

[MGCL07] Daniel Mateus, Jean-Pierre Gallois, Jean-Paul Comet and Pascale Le Gall, Symbolic modeling of genetic regulatory networks, Journal of bioinformatics and computational biology, Imperial College Press 05, 627 (2007)

[NDB14] Maxime Neyret, François-Xavier Dormoy et Jean-Christophe Blanchon, Méthodologie de validation des spécifications fonctionnelles du contrôle-commande, Application au cas d'étude du Système de Réfrigération Intermédiaire (SRI), Génie Logiciel Hors-série « L'initiative Connexion : IDN et Contrôle-Commande »: 12-25, 2014.

[RGLG03] Nicolas Rapin, Christophe Gaston, Arnault Lapitre, Jean-Pierre Gallois Behavioural Unfolding of Formal Specifications Based on Communicating extended automata ATVA 2003

[TD03] Travé-Massuyès Louise, Dague Philippe, Modèles et raisonnements qualitatifs ; TraitéIC2, série Systèmes automatisés, ISBN : 9782746207448, 2003.

[TK02] Tiwari, A., and Khanna, G. (2002). Series of abstractions for hybrid automata. In Hybrid Systems: Computation and Control, LNCS 2289, 465-478, Springer.