Qualitative Models for the Supervision of CPS Simulations

Baptiste Gueuziec
Jean-Pierre Gallois
baptiste.gueuziec@cea.fr
jean-pierre.gallois@cea.fr
Université Paris-Saclay, CEA, Institut LIST
Gif-sur-Yvette, France

Frédéric Boulanger
frederic.boulanger@centralesupelec.fr
Université Paris-Saclay, CNRS, CentraleSupélec,
Laboratoire Méthodes Formelles
Gif-sur-Yvette, France

ABSTRACT

The simulation of complex systems is an activity that affects many industrial and research fields and is important to verify the future behavior of a system. When it comes to hybrid systems [1], the heterogeneity of the discrete and continuous parts makes the modeling and simulation more difficult. Usually, numerical methods are chosen to simulate the continuous part of the models. However, their cost can be high when precision is needed and when interactions with the discrete part force rollbacks in the simulation. Other techniques are proposed to deal with uncertainties and expensive models, such as the flow-pipe methods [2] or qualitative modeling [3]. In this article, we present preliminary work to rely on a qualitative analysis of the model to supervise its simulation. Our goal is to build a qualitative map of the state space of the model in order to adapt the quantification of values and the discretization of time of the integration method. We expect this supervisor to optimize the time/precision balance, especially in the case of complex systems with many components.

ACM Reference Format:
Baptiste Gueuziec, Jean-Pierre Gallois, and Frédéric Boulanger. 2024. Qualitative Models for the Supervision of CPS Simulations . In *Proceedings of MoDeVVa 2022 Workshop (MoDeVVa 2022).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3550356.3561594

1 INTRODUCTION

Quantitative methods, such as numerical integration, give precise results but may consume a lot of time and resources. Qualitative techniques use a symbolic interpretation of the models without knowing all the numerical parameters by relying on dependency relationships between variables. They are less precise but can be applied early in the design phase. They can be used to plan numerical simulations according to the objectives and improve the results of analyses (proofs, optimization, diagnosability, etc.) [4].

Brown [5] and De Kleer [6] first mentioned qualitative representation of information in their works related to computer-assisted calculus for physics, such as electronics and classical mechanics. They created an opposition between qualitative and quantitative knowledge and proposed the idea of a causal ordering of events. Hayes [3, 7] pursued this idea and introduced the concepts of qualitative modeling and naive physics. Forbus developed the concepts further in a more complex theory of the process [8] and paved the way for the modern qualitative modeling theories. Kuipers [9] and Tiwari [10] proposed two major approaches: the former studied a group of models described by causality relations and qualitative differential equations, while the latter proposed a division of the hybrid systems based on a description by ordinary differential equations. Previous work in our teams followed Kuipers' approach [11], while others followed Tiwari's [12, 13].

In qualitative modeling, the dynamics of complex systems are described as modes, phases, and events [8]. Each mode is defined by a set of differential equations [10]. A phase is a period of time that corresponds to a particular set of results of these equations and is associated with a specific evolution of the state variables. Events are instantaneous occurrences that mark a transition between modes or phases. Complex systems are often heterogeneous i.e. with many modes and, therefore, many kinds of dynamics, requiring discrete and continuous variables to be managed in parallel. However, since this article deals with preliminary work, we will focus on the simulation of the continuous part of a system and choose the Brusselator as a case study to illustrate our approach.

2 QUALITATIVE MODEL OF THE STATE SPACE

The Brusselator, first described in [14], models a chemical reaction between two components, whose concentrations are noted x and y, and are characterized by the following ordinary differential equations with parameters a and b :

$$\begin{cases} \dot{x} = 1 - (b+1)x + ax^2y \\ \dot{y} = bx - ax^2y \end{cases}$$

Starting from the initial concentrations x_0 and y_0, and knowing the parameters a and b, we want to determine the kind of behavior of the concentrations. A numerical analysis of the Jacobian [14] proves that if $b < a + 1$, the system will follow a pseudo-cycle converging to the point $(x = 1, y = \frac{b}{a})$. Otherwise, it will generate a non-converging cycle. One usually computes the system's evolution with a numerical integration method, such as Euler or Runge-Kutta. To save computation time and focus only on visualizing specific properties such as the periodicity, previous work in our team [12, 13] relied on the partitioning of the state space with the nullclines of the two derivatives $\dot{x}$ and $\dot{y}$. These nullclines are computed by finding the zeros of the differential equations, which gives y as a function of x in the following functions $f_{\dot{x}} : x \to \frac{(b+1)x-1}{ax^2}$ and $f_{\dot{y}} : x \to \frac{b}{ax}$. The computed partition splits the state space into phases defined by the direction of variation of the two state variables (see Figure 1). The relative position of the current state to these nullclines gives the direction of variation of each variable. We represent this information with the black arrows in the figure.

MoDeVVa 2022, Montréal, Canada,

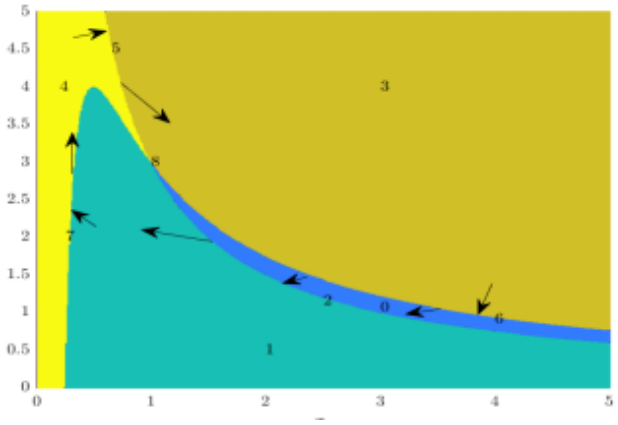

Figure 1: Qualitative state space for a Brusselator

This method is efficient when we seek only general and imprecise information about the model's behavior. To obtain more details, we intend to mix these qualitative possibilities with quantitative simulation to optimize the state space partition, the quantification of the value of state variables, and the adaptation of the integration step. This should allow for better use of the computation power where it is relevant.

2.1 Isoclines as neighborhood borders

The first step in the supervision of the numerical simulation is to refine the partition of the state space. For this, we divide the phases identified thanks to the nullclines into areas according to the proximity of the current state to a potential event.

For instance, in our example, the nullclines delimit four phases numbered 0, 1, 3, and 4 in Figure 1, and define four events numbered 2, 5, 6 and 7. A fifth event (numbered 8) is the convergence point. Events correspond to transitions between phases. To build a finer abstraction of the state space structure, we introduce isoclines (lines of constant derivative), which give information about the proximity to a nullcline or an event.

For example, we can define our isoclines with $y = \frac{(b+1)x-1 \pm c_1}{ax^2}$ and $y = \frac{bx \pm c_2}{ax^2}$, where c_1 and c_2 are constants that depend on how fine or coarse we want the model to be. If we place an isocline at the same distance on each side of a nullcline, they will delimit a neighborhood for the nullcline. If we consider the previous partition of the state space as a separation between phases (defined by the dynamics of the system), we now have a separation into areas defined by their proximity to a border between phases. This allows us to identify "safe" areas in which we know that no border will be crossed during the next simulation step, and proximity areas, in which we should adapt the simulation policy to take care of possible phase changes and prevent rollbacks in the numerical integrator. We now have a topographic map of the state space that will help us to explore the system's behavior with a finer division of the state space, which is encouraged in qualitative studies [10].

2.2 Quantization

The previous division of the state space along nullclines and isoclines depends only on a qualitative analysis of the behavior of the system. However, since our goal is to supervise a quantitative simulation of the system, we have to divide the state space into rectangles by quantizing the state variables. For now, we consider a regular quantization that does not depend on the value of the state variables. We will see later that during the simulation, we will have to adapt this quantization depending on the value of the integration step. A purely qualitative model of the state space is usually finite (there is a finite number of phases). A purely quantitative version is usually infinite and non-countable (assuming real values for the state variables). Our quantized model of the state space can be considered as semi-qualitative and may not be finite but is countable.

3 SUPERVISED QUANTITATIVE SIMULATION

Guided by our qualitative map of the state space, we can supervise the numerical simulation of the model.

3.1 Adaptive Time Steps

To reduce the number of simulation steps, we will use an adaptive integration step for computing the value of the state variables over time. Some programs make a first simulation to get an approximation of the result and make a second run with an adapted integration time step [15]. We prefer a single-run method, which is suitable for real-time analysis. We adapt the integration step to the speed of variation of the variables in order to limit numerical errors, and we adapt it according to our semi-qualitative model of the system to determine precisely when events occur.

3.2 Time Step and Maximum Variation

The usual way to adapt the integration step to the speed of variation of the state variable is to use the largest integration step that guarantees a bounded error on the computed value. To simplify the computations, we chose to reason with decimal orders of magnitude, i.e. $\lceil \log_{10}(|h\dot{x}|) \rceil$ should be constant, with h the current integration step. In our case study with two variables, we consider the maximum magnitude of the two derivatives to determine h. We use magnitudes of 10 to stay close to human reasoning, which is key to qualitative modeling [16]. We also did experiments with bases such as 2 and 16. It appears that using smaller bases gives better precision and that base 2 is more suitable for numerical computations. However, base 10 makes the program structure more intuitive and understandable. The choice between these possibilities will depend on the objective of the model and the preference between efficiency and simplicity. Bigger bases for the order of magnitude can be considered on models with potential significant variations.

The constant value of $\lceil \log_{10}(|h\dot{x}|) \rceil$ must be chosen accordingly to c_1 and c_2, which determine the safe distance from the nullclines. In order to prevent missing any transition between two phases, it should be impossible to change of phase in one step without hitting the neighborhood of the nullcline. This criterion ensures that the system will not have an unpredicted transition, which would require a rollback of the simulation to detect its exact spot.

3.3 Proximity Planning

The second adaptive aspect we use in our approach is the adaptation of the integration step to the distance to the nullclines. This is where the isoclines we introduced earlier play a central role. We aim to avoid missing any transition between phases or computing a transition that does not exist by using a bad integration step.

Therefore, we reduce the integration step by a given magnitude when the current state enters the neighborhood of a nullcline, defined by the isoclines. We chose to use a magnitude of 10 for the same reason we evoked in the precedent paragraph. This reduction of the integration step is even more important in areas that are close to all nullclines. If the total derivative is noted $\overrightarrow{d} = \dot{x}\overrightarrow{dx} + \dot{y}\overrightarrow{dy}$ and if $\dot{x}$ changes its sign at instant t, by continuity and because of our adaptation of h to the derivative, $\dot{x}_{t-h}$ and $\dot{x}_t$ will be both in a close neighborhood of the nullcline $\dot{x} = 0$. So if we are not at the same time in a close neighborhood of the nullcline $\dot{y} = 0$, then $\dot{x} = o(\dot{y})$ and $\dot{x}\overrightarrow{dx} + \dot{y}\overrightarrow{dy} \approx \dot{y}\overrightarrow{dy}$. However, if we are in the neighborhood of the $\dot{y}$ nullcline, then a change in the sign of $\dot{x}$ may strongly affect the total derivative.

To avoid false transitions, our integration step reduction must be additive, and the system must reduce the integration step even more when in the neighborhoods of multiple nullclines.

3.4 Variable quantification

We quantized the state space of the system when building the semi-qualitative model that we use to supervise the quantitative simulation. However, our experiments showed that we need more information and smaller quantization rectangles when we reduce the integration step. Otherwise, we accumulate errors too fast and drift too far from the real trajectory. A solution is to adapt the quantization to the current time step. For this, we use a function taking as parameters the initial state space precision and the current integration step and returning the current quantization. Thinking in terms of order of magnitude, we choose the size of the quantization rectangle as $d = h \times d_i$, with d the current quantization step, d_i the initial one, and h the current integration step.

4 RESULTS

In this section, we apply our approach to a Brusselator and compare the results to other methods. We made many simulations starting from different initial points, but we show the results starting from the $(5, 4)$ initial point. Other starting points gave similar results. We stop the simulation at time 10. For the quantitative simulation, we used the Euler method. Everything was implemented in Python. For all the plots shown here, abscissas and ordinates correspond respectively to the values of x and y.

4.1 Fully adaptative vs. fixed integration step

First, we choose $a = 1$ and $b = 1.7$ to make the system convergent. The initial integration step is set to $h = 0.1$. Our results are pretty close to the actual behavior computed with precise numerical methods (see Figure 2). In contrast, a fixed-step model with a time step of 0.1 diverges completely from the real solution, and has already left the validity area of the differential equation with a negative

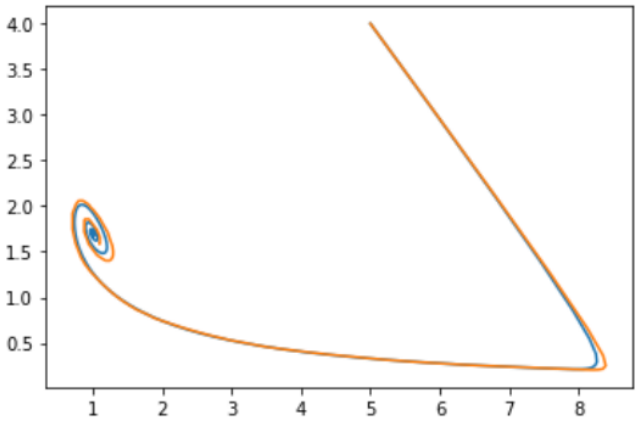

Figure 2: Our results (orange) vs. real behavior (blue)

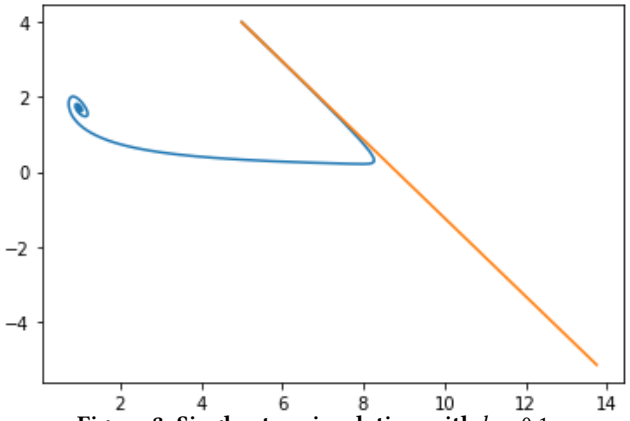

Figure 3: Single-step simulation with $h = 0.1$

value of y after only one step (see Figure 3). When pushed further, this simulation diverges to give meaningless results.

To obtain results similar to those given by our approach, we have to use a fixed integration step $h = 0.01$. However, this leads to the computation of 1002 steps in 0.08 seconds, while our approach computes only 290 steps in 0.07 seconds.

To illustrate the importance of that difference, we show the same kind of results with $a = 1$ and $b = 2.7$, which makes the system divergent, and change the stopping criterion to a fixed number of steps. The results are shown in Figure 4 and Figure 5. This example shows that our approach computes an entire cycle of the behavior in 280 steps. On the contrary, the fixed integration step method requires more than 800 steps as shown in Figure 6.

The difference with our approach is particularly visible when the values stay far from the nullclines. In these areas, our semi-qualitative model efficiently supervises the numerical simulation, taking advantage of the fact that it is useless to keep a small time step when there is no chance of having some brutal change in the derivative or in the values of x and y. This saves time and execution resources that can be better used in critical areas where they are needed.

On a simple system like the Brusselator, dividing the total number of simulation steps by more than three does not yield a big

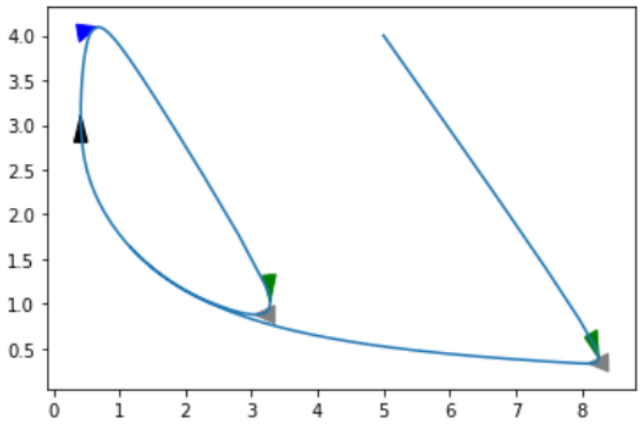

Figure 4: Our approach on 280 steps

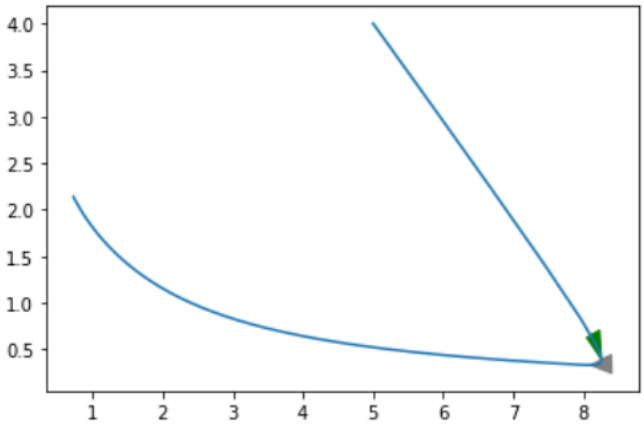

Figure 5: Fixed integration step (0.01) on 280 steps

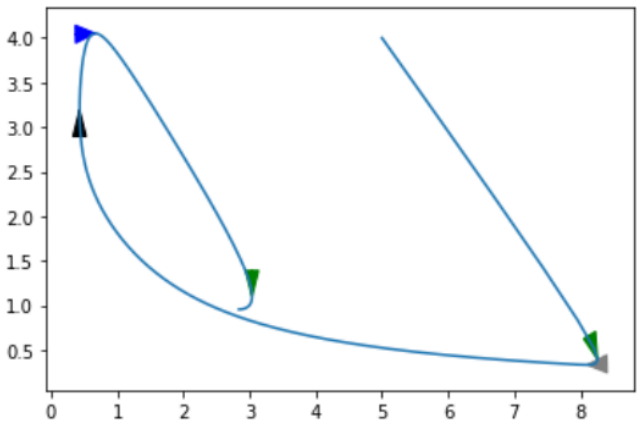

Figure 6: Fixed integration step (0.01) on 800 steps.

improvement in execution time. However, on complex systems with many components, where each step is more costly, we believe that the improvement will be more impressive and that the extra computations we perform to supervise the simulation will save more time than they consume. This has still to be verified on a more significant use case.

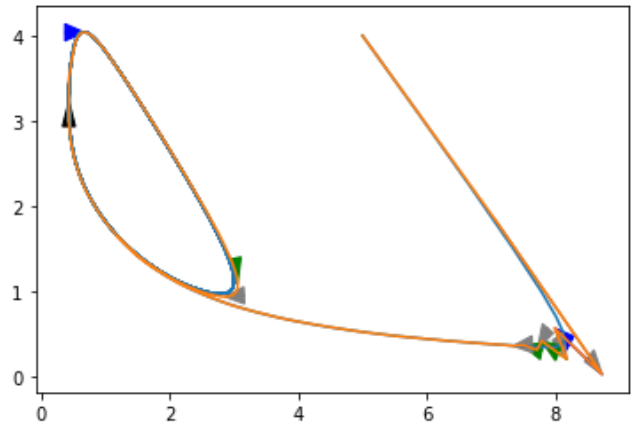

Figure 7: Fixed integration step of 0.025 and false transitions

To complete the comparison, we show how the fixed integration step method generates false transitions when it gets near the two nullclines. Figure 7 shows the result of the simulation with a fixed integration step of 0.025 on one full cycle of the system. Transitions are represented as colored arrows on our figures. We can see that when we get near the two nullclines, there are five false transitions between phases that should not exist. This is easily understandable considering that when both derivatives have a low absolute value, a change of sign completely reverses the total derivative of the system. On the contrary, when one of them has a high absolute value, a change of sign of the lower one does not significantly influence the total derivative. Yet, the inability to react quickly in the presence of a sufficiently high derivative causes another deviation.

4.2 Partially adaptive models

To better understand the role of the two kinds of adaptation of the integration step we use in our model, we ran the simulation with partially adaptive models.

Figure 8 shows the result of a full-cycle simulation when the integration step (initially set to 0.025) is only adapted to the proximity of the nullclines. Significant errors appear very quickly when the state variables vary rapidly but are corrected progressively when the derivatives are low. Hence, the approximation of the limit cycle is still quite good and we observe three false transitions.

Figure 9 shows the result of a full-cycle simulation when the integration step (initially set to 0.025) is only adapted to the variation rate of the variables. The approximation is better in the area where the variables evolve rapidly. Still, we observe as many false transitions as before, due to the lack of precision near the two nullclines.

This shows that the combination of both adaptations is required to get good results with a reasonably sized initial integration step.

5 PERSPECTIVES

5.1 Multiple isoclines

We saw the interest of having an isocline on each side of a nullcline to delimit an area in which we take great care of transitions

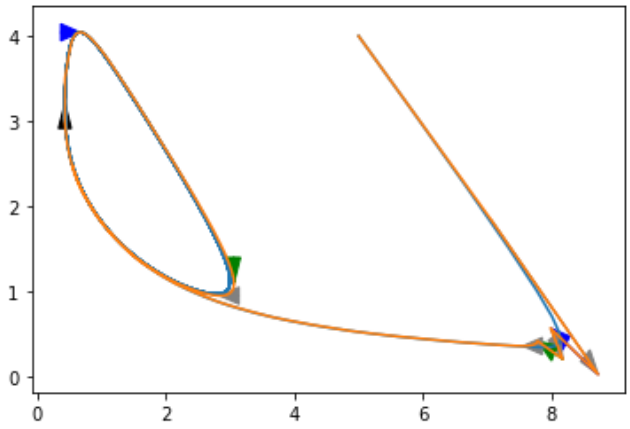

Figure 8: Proximity adaptation only.

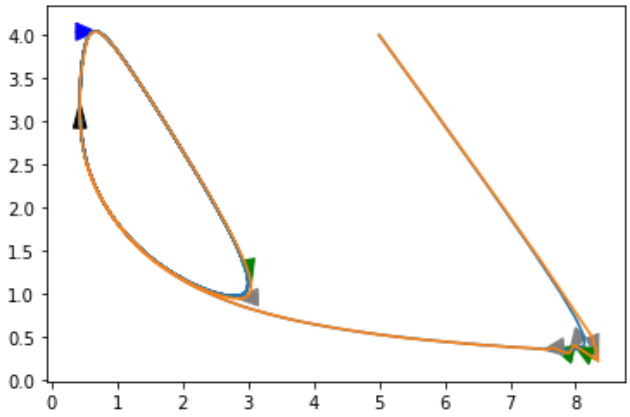

Figure 9: Variation adaptation only.

between phases. However, Figure 8 shows that the adaptation of the integration step to the proximity of the nullclines is insufficient to guarantee the precision of the value of the state variables. We could therefore think that using more isoclines to have a more progressive adaptation to the proximity of phase changes is not useful. However, the adaptation to the proximity of the nullclines gives more accurate locations for the transitions. In a real-time context, for the control of a cyber-physical system, additional isoclines could therefore help to anticipate the transitions of the system and allow for an efficient adaptation of the sampling rate of the continuous part by the controller.

5.2 Thick borders

Because of the quantization of the value of the state variables, the computed state of the system is never *exactly* on a nullcline, so we detect events only when we have crossed it. The adaptation of the integration step avoids overshooting the phase change too much, and using multiple isoclines improves the adaptation and makes it smoother. However, making the border a bit thick around the nullcline could help to identify transitions, and it would also help to avoid jumping back and forth over the nullcline when the behavior of the system remains in its close neighborhood. The main issue with this idea is to choose the right thickness for the border: if it is too thin, we may compute too many transitions that do not exist, and if it is too thick, we may miss significative transitions of the system.

6 CONCLUSION AND FUTURE WORKS

The results presented in this article show that there is an interest in building a qualitative model of a system and using semi-qualitative logic to supervise a numerical simulation to optimize the resources consumed by the simulation of the system when verifying its behavior. We obtained these results on a continuous model, but the improvement in the detection of phase changes shows that our approach can improve the simulation of hybrid systems and the control of cyber-physical systems.

In this preliminary work, we focused the qualitative analysis on the structure of the state space, but other modeling paradigms could be used to improve the supervision of numerical simulations. For instance, using models in the spatial (or time) frequency domains could also help to anticipate the system's behavior.

Another possible application of this approach is to explore the design space for a family of systems instead of the state space of one system. For our use case, we would build qualitative models that describe how the behavior of the system varies depending on the parameters a and b. This would reduce the number of simulations required to determine safe or optimal values for the parameters of the system.

REFERENCES

[1] Thomas A Henzinger. The theory of hybrid automata. In *Verification of digital and hybrid systems*, pages 265–292. Springer, 2000.
[2] Olivier Bouissou, Alexandre Chapoutot, and Samuel Mimram. Computing flowpipe of nonlinear hybrid systems with numerical methods. *arXiv preprint arXiv:1306.2305*, 2013.
[3] Patrick J Hayes. *Readings in qualitative reasoning about physical systems*, chapter The second naive physics manifesto, pages 46–63. Morgan Kaufmann, 1985.
[4] Philippe Dague Louise Travé-Massuyès. *Modèles et raisonnements qualitatifs*. Hermes, 10 2003.
[5] Allen L Brown. *Qualitative knowledge, causal reasoning, and the localization of failures*. MIT Artificial Intelligence Laboratory, 1974.
[6] J. De Kleer. *Qualitative and Quantitative Knowledge in Classical Mechanics*. AI-TR-. Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1975.
[7] Patrick J Hayes. The naive physics manifesto. *Expert systems in the microelectronic age*, 1979.
[8] Kenneth D Forbus. Qualitative process theory. *Artificial intelligence*, 24(1-3):85–168, 1984.
[9] Benjamin Kuipers. Qualitative simulation. *Artificial intelligence*, 29(3):289–338, 1986.
[10] Ashish Tiwari and Gaurav Khanna. Series of abstractions for hybrid automata. In *International Workshop on Hybrid Systems: Computation and Control*, pages 465–478. Springer, 2002.
[11] Slim Medimegh. *Analyse formelle de spécifications hybrides à partir de modèles SysML pour la validation fonctionnelle des systèmes embarqués*. PhD thesis, Université Paris Saclay (COmUE), 2018.
[12] Jean-Pierre Gallois and Jean-Yves Pierron. Qualitative simulation and validation of complex hybrid systems. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.
[13] Hadi Zaatiti. *Modélisation et simulation qualitative de systemes hybrides*. PhD thesis, Université Paris-Saclay (ComUE), 2018.
[14] Shaun Ault and Erik Holmgreen. Dynamics of the brusselator. *Math 715 Projects (Autumn 2002)*, 2, 2003.
[15] Brian Stout. Méthodes numériques de résolution d'équations différentielles. *Marseille, France: Universite de Provence*, 2007.
[16] Barry Smith and Roberto Casati. La physique naïve: un essai d'ontologie. *Intellectica*, 17(2), 1993.