Qualitative Reasoning and Cyber-Physical Systems: Abstraction, Modeling, and Optimized Simulation

Baptiste Gueuziec
Université Paris-Saclay
CEA, List
F-91120 Palaiseau, France
baptiste.gueuziec@cea.fr

Jean-Pierre Gallois
Université Paris-Saclay
CEA, List
F-91120 Palaiseau, France
jean-pierre.gallois@cea.fr

Frédéric Boulanger
Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire Méthodes Formelles
F-91190, Gif-sur-Yvette, France
frederic.boulanger@centralesupelec.fr

***Abstract*—Complex systems modeling and simulation are critical in many industrial and research fields and specifically to predict, prove, verify, and understand the behavior of cyber-physical systems. The diversity of variables in a system creates complexity [1] and a need for more efficient modeling and simulation methods. In the case of hybrid systems [2], the heterogeneity of the discrete and continuous parts makes these tasks more challenging by adding the necessity to manage different types of variables and transitions separately. Qualitative reasoning offers a paradigm to study the behavior of such systems with a high level of abstraction, trading precision and specificity against generality and formalism. This paradigm can be preferred to numerical analysis in specific situations, especially in the upstream study of a system, in its design phases. However, the different representations and the various contexts of such systems create an important obstacle to define a general methodology for applying qualitative reasoning and modeling to every case. In this article, we propose a method and a tool to unify different qualitative reasoning techniques on complex cyber-physical systems. We will also develop the possibilities offered by the obtained abstraction in various tasks such as formal proof, verification, property analysis, diagnosis, simulation driving, and system monitoring.**

***Index Terms*—modeling, qualitative reasoning, cyber-physical systems, abstraction, simulation**

I. Introduction

Cyber-physical systems [3] are central to many disciplines. The ability to study, decompose, and understand them and their behavior is crucial to increasing our technical abilities in many industrial and scientific areas. Many methodologies exist to study them, either at a component level or from a more systematic point of view. However, all these methods lack generality: they were mainly thought for precise applications, be it a numerical simulation, test, or monitoring. Their high degree of specificity makes them sparsely adapted to tasks that require more general and abstract properties, and to reason on the system. Moreover, the hybrid nature of many CPS including feedback or human interactions adds a new form of complexity. As hybrid systems [2] combine discrete and continuous variables, their study is a more challenging task and must involve more efficient and general methods.

Definition: We consider as an hybrid system any system containing both continuous and discrete behaviors. These systems can be described with:

- a discrete variable Q, also called the *mode*, defined on the finite domain $\mathbf{Q}$ and whose value is noted m,
- a set X of continuous variables defined on the continuous set $\mathbf{X}$, with value x. The components of X are noted X_i, with value x_i. In view of practicality, we often consider $\mathbf{X} = \mathbb{K}^n$ with $n = |X|$.

We define a variable as a measurable quantity, with a physical unit and a value. We also integrate into our description a function I mapping each mode to a set of predicates $Inv_m \subset 2^{\mathbf{X}}$ representing the invariants of the system for the mode m; a set $T \subset \mathbf{Q} * 2^{\mathbf{X}} * \mathbf{Q} * \mathbf{X}^{\mathbf{X}}$ of modal transitions represented by the departure mode m_1, a transition condition (a guard), the target mode m_2, and a reset function giving the new value of X when entering m_2; and the flow F mapping every mode m of Q to the dynamics vector of X containing ordinary differential equations (ODE) on X and its derivatives.

Originally, qualitative modeling was introduced by Brown [4] and De Kleer [5], who developed the concept of qualitative knowledge about systems and processes. They applied this theory mainly to electronics and computer-assisted physics computation. They did not design this representation of knowledge as a substitute for numerical computation but as a complementary strategy. Indeed, they presented their approach as a tool to solve general problems by reasoning at a high level of abstraction on the system and to refine the more specific sub-problems that could only be solved with a more classic numerical computation. Therefore, the initial idea was to add intelligence in problem-solving and optimize the use of computational resources that should be kept for sub-problems that really require them. If the notions implied by qualitative have evolved since then, the ambition remains unchanged: proposing a new reasoning tool to supplement the very precise but too specific numerical approaches available to study the different kinds of systems. Major elements have been added to the theory of qualitative modeling, such as naive physics [6], [7], the theory of dynamic processes [8], and the concept of conceptual closure of such a theory [9]. Qualitative modeling has made significant advances with the works of Kuipers [10] on the qualitative representation of the state space and the value of system variables. He introduced a form of reasoning based on

sign algebra, using the values $\{-, 0, +\}$ as abstractions of the numerical values of the variables. The authorized operators are $\{+, *\}$, and they illustrate the advantage of the sign algebra as they preserve all their properties of transitivity, associativity, and commutativity [11]. This method allows the qualitative study of the behavior of a system based on qualitative differential equations, which are abstractions of numerical differential equations. The development of this analysis led to the development of the **QSIM** tool. However, the non-determinism of such operations and the lack of precision showed the limits of the approach in the case of systems with feedback or multiple successors for a given state. Kuipers and Berleant partially resolved these problems with their work on semi-qualitative reasoning [12]. In this approach, the sign algebra is completed with interval propagation to integrate a part of numerical analysis and resolve the uncertainties that cannot be studied with only sign knowledge. This complementarity allowed the study of more complex systems and the development of more advanced versions of his tool, such as **SQSIM** and **Q3**. Some work has been undertaken to combine it with orders of magnitude [13], but the results did not meet the expectations. However, interval propagation has since really progressed, allowing advances such as flow-pipe computation [14] or more precise uncertainty measures and correction. Tiwari completed this approach [15] and generalized it to ordinary differential equations (ODE) under the condition that the terms of the equations must be polynomial according to the system's variables. This methodology requires an additional step of numerical analysis upstream of the system study. However, it can give more interesting results as it takes into account the links between the dynamics and the values of a system, while previous methods had the drawback of separating the two aspects. In terms of applications, qualitative reasoning mainly allows formal computation and proof using temporal logic [16], diagnosis [17], [18], verification [19], [20] and approximation [21]. Qualitative modeling implies reasoning on multiple levels of abstraction. The most important ones are the design space (also named 'domain space' by Forbus [9]), centered on constructing an adapted representation of the system and its context. Working on the design is the most advanced form of qualitative reasoning as it involves non-instantiated and even partially defined systems and therefore requires the ability to compute and solve equations with purely symbolic and non-valuated constants. The dynamic equations of such a system can, for example, be defined as $\dot{x} = ax^3 - 3xy^2$ with a a symbolic constant with unknown value. The stakes of such a challenge are to help in the design of complex systems very upstream in their development, and to work on models defined with incomplete knowledge. The state space of a system is a more tangible field, as we can expect all the fundamental knowledge and values of the system to be at least partially known. In this publication, we will focus on the exploration and exploitation of the state space, but we consider these results as preliminary to further works to explore both design and state spaces. This article will present the current state of our methodology and tool to create a qualitative model from an expression of the system, explore it and create its behavior tree, and exploit it for behavior analysis, simulation piloting, or system monitoring.

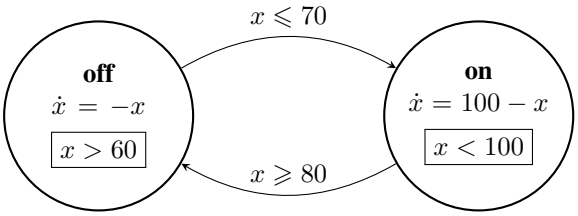

Fig. 1. Hybrid model of a thermostat

II. Concepts

A. Cyber-Physical Systems

The main application of our works on qualitative modeling is the analysis of cyber-physical systems. We include in this term every system consisting of physical elements controlled by a computational process. The representation we choose to use as a reference takes the form $S = \langle Q, X, I, F, (m_{init}, x_{init}), T\rangle$ with Q the discrete mode defined on $\mathbf{Q}$, X the vector of continuous variables defined on $\mathbb{K}^n$ with $\mathbb{K} = \mathbb{R}$ in many cases, F the flow of the system (i.e., its dynamics), which maps the value m of Q to the set of ODE that define the dynamics of X in mode m, m_{init} the initial mode and x_{init} the initial value of X. I is defined as explained above and represented as a mapping from $\mathbf{Q}$ to predicates on $\mathbf{X}$. T is the set of all modal transitions, represented as quadruplets (initial mode, transition condition, arrival mode, reset values), where the reset values are represented with a function in $\mathbf{X}^{\mathbf{X}}$ that gives the value of the continuous variables after the transition from their values before the transition. The equations of F are generally limited to $\mathbb{K}[X_i]_{X_i \in X}$ (we will use the notation $\mathbb{K}[X]$ for the rest of this article), the set of polynomials on $\mathbb{K}$ with variables in X. As F defines the continuous behavior of the system for each mode and T the condition to switch the current mode to its successor, we can now model the system's operation.

For example, let us define a model of a thermostat as:

$Q = \{mode\}$,
$\mathbf{Q} = \{\text{off}, \text{on}\}$,
$X = \{x\}$ with x the temperature of the system in Celsius,
$\mathbf{X} = \mathbb{R}$,
$I = \{mode = \text{off} : \ x > 60, \ mode = \text{on} : \ x < 100\}$,
$F = \{mode = \text{off} : \dot{x} = -x, mode = \text{on} : \dot{x} = 100 - x\}$,
$(m_{init}, x_{init}) = (\text{off}, 80)$,
$T = \{(\text{off} \xrightarrow{x \leqslant 70} \text{on}, x \mapsto x), (\text{on} \xrightarrow{x \geqslant 80} \text{off}, x \mapsto x)\}$

B. Hybrid Automaton

Hybrid automata [2] are a standard and efficient tool to represent and study cyber-physical systems. They can be defined as $H = \langle Q, X, V, E, Init, I, F, J, L\rangle$ with Q, X, I,

and F corresponding respectively to the set of discrete variables, the set of continuous variables, the invariant constraints, and the flow equations as for cyber-physical systems. (V, E) is the control graph of the automaton, with V its vertices representing all its states (in our syntax, V rather represents the modes) and E the edges between the vertices. To represent a cyber-physical system as an automaton, E can be obtained by analysis from the transitions in T. *Init* is the set of initial conditions, with predicates constraining the initial values of Q and X. L is a pair composed of a set of labels and a mapping from E to these labels. The label associated with a transition can be the nature of the transition, its cause, or its specificity. The structure of hybrid automata is particularly adapted to represent the behavior of a cyber-physical system. For example, we represented the thermostat system described above as an hybrid automaton in Figure 1. This classic definition isolates the different modes of the system as different states of the automaton. Each one is characterized by its specific flow conditions (i.e., the continuous dynamics associated with the mode) and by transitions and initial conditions, defining the values of the elements of X that allow a transition to another mode and the reset values imposed after an incoming transition. However, the trajectory of X is not taken into account in this automaton model. In simple cases like this one, it can be deduced from the expression of the flow, whereas it can be impossible for more complex systems. This means that this structure is oriented toward studying discrete behaviors rather than continuous ones. However, studying CPS also requires observing the continuous behavior of the system. To this extent, we must not only deal with the dynamics of the continuous variables but also with their trajectories and properties, as these elements can give us helpful information about the behavior of the global system. Moreover, knowing the behavior of the continuous variables is necessary to know which continuous transitions (constrained by the value of X) can occur, and which ones are impossible in the current mode. To this extent, we need to be able to compute the trajectory of these variables.

One can compute these trajectories at two levels of abstraction, with two main families of techniques. The first one, and the most commonly used, is the family of numerical methods. Here, the trajectories are obtained with numerical simulation techniques, including, for example, Euler integration or Runge Kutta, that can give exact and reliable results. The main problem of numerical methods in our context is that they do not allow any generality in the simulation and may require very high computation time for complex systems. Moreover, any uncertainty can propagate and give very unpleasant results. Uncertainty management needs to use more complex techniques such as interval propagation methods, which require more time and computational resources. Therefore, as we especially seek an important generality more than precision at this step of the system study, we prefer the other option, which is qualitative reasoning. This choice implies discretizing the state space in a finite number of qualitative states, which are abstractions of the numerical values they include. This discretization will allow the computation and study of an abstract trajectory among the set of qualitative states. These trajectories will benefit from being very general and represent a whole family of numerical traces as a type of behavior. We will represent such trajectories in another type of hybrid automata featuring both modal and intra-modal transitions. From now on, we will present how to obtain these trajectories.

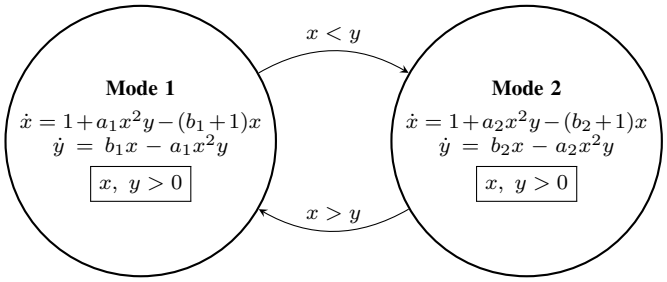

Fig. 2. Hybrid model of a Brusselator system

Definition: In a qualitative model, we consider **discrete** or **modal** transitions as changes in the values of the discrete variable Q. The **intra-modal** or **qualitative** transitions correspond to a change of the qualitative state without change of mode.

III. System Abstraction

A. Abstraction Methods

The first step to compute the qualitative trace of a system is to transform our representation of the system to a qualitative model to serve as computation support. Let us consider a system $S = \langle Q, X, I, F, S_0, T \rangle$ with the notations already defined.

We consider that the flow of the system is represented by a mapping of each mode to an ordinary differential equation $\dot{X} = F(X(t), t)$ with t the time parameter of the system.

To illustrate our explanations, we will take as an example the hybrid Brusselator system, expressed as:

$$
\begin{aligned}
S \;=\; & \langle \\
& Q \;=\; \{mode\},\ X = \{x, y\}, \\
& \mathbf{Q} \;=\; \{1, 2\},\ \mathbf{X} = \mathbb{R}^2, \\
& I \;=\; \{mode = 1 : x, y > 0;\ mode = 2 : x, y > 0\}, \\
& F \;=\; \{mode = 1 : \\
& \qquad (\dot{x}, \dot{y}) = (1 - (b_1 + 1)x + a_1x^2y, b_1x - a_1x^2y), \\
& \quad mode = 2 : \\
& \qquad (\dot{x}, \dot{y}) = (1 - (b_2 + 1)x + a_2x^2y, b_2x - a_2x^2y), \\
& S_0 \;=\; (1, (5.3, 2.6)), \\
& T \;=\; \{(1, x < y, 2, Id),\ (2, x > y, 1, Id)\} \\
& \rangle
\end{aligned}
$$

With this representation, it is possible to visualize the discrete part of the model as a hybrid automaton, as already explained (see Figure 2). For the continuous evolution, different abstraction methods have been introduced and used. The choice of the abstraction method is critical because this will mainly influence the possibilities and the precision of the state-space exploration. Among the methods that have been studied and proposed, the more useful in the case of

CPS are the methods of Kuipers and Tiwari. The use of more advanced reasoning techniques implies converting the continuous trajectory into a discrete evolution with the help of an abstraction function α, that associates each position of the system to a qualitative state, which belongs to a finite set. The nature of this function is what differentiates the various modeling approaches. The first method, brought by Kuipers, only reasons in terms of landmarks (i.e., hyperplanes defined by $X_i = c$ with c a constant and $i \in [\![0, |X| - 1]\!]$). The abstraction of the variation space is made separately for the variables themselves and their derivatives. The abstraction of the state-space is done using landmarks on the zeros of the variables and abstractions of the system's differential equations named qualitative differential equations (QDE) to make any change of sign have influences on the other variables. For example, if $X = (x, y)$ and if x has a positive influence on y, then the QDE characterising it will be $y = M^+x$. In the case of our case study system, $\dot{x} = 1 - (b + 1)x + ax^2y$ will be replaced by $\dot{x} = M^+y + M^+x * M^+y$. As variables and their derivatives are not completely linked anymore because of the high level of abstraction, using these landmarks on the components of $\dot{X}$ is more complex and does not give much information. Actually, $\dot{x} = ay$ and $\dot{x} = ay^2$ are not different in this abstraction space applied to $\mathbb{R}^+$.

It is also possible to use other landmarks (i.e., with $X_i = c \neq 0$) deduced from prior knowledge about the system. For example, if we know that 100 $km.h^{-1}$ and 1 $km.h^{-1}$ are important milestones around which the qualitative reasoning about an autonomous car should process, they will be considered as reference landmarks to compare and abstract the current value of the speed. However, as these values are not from the sign algebra, they will not be propagated in the equations as their properties do not match all the properties of the most convenient algebra. This method allows simple studies of systems based on explicit values chosen according to the objectives and the context of the CPS. It means that it is required to have a predefined instance and context of the system and to know where and why it will be used. This constraint is contrary to the main objective of qualitative modeling: we must create models with as little information as possible, so we should avoid contextual frontiers.

The approach of Tiwari has resolved this drawback: for each mode $m \in \mathbf{Q}$, using the equations defining the system (both the dynamic equations, the invariant expressions, and the transition conditions), the algorithm defines new variables derived from X. Using all the elements p from F_m, I_m, and T_m (with F_m, I_m, and T_m being the subset of F, I and T associated with mode m) such that $p \in \mathbb{K}[X]$, we define a set P_m initiated with $P_m = \{X_0, \ldots, X_{n-1}\}$ with $n = |X|$. $\forall p \in \mathbb{K}[X] \cap (F_m \cup I_m \cup T_m)$, we set $x_p = p$ and we add each x_p to P_m. Then, $\forall p \in P_m$, if $\dot{p} \neq 0 \ \wedge \ \dot{p} \notin P_m \ \wedge \ \nexists (b, d) \in \mathbb{K}[X] * P_m$ such that $\dot{p} = b * d$, then $\dot{p}$ is added to P_m. As these polynomials include terms only defined by their differential equation, it is likely that many of them are neither nilpotent nor idempotent. Therefore, the more they are derived, the more complex their expression will become. For example, in the case of the Brusselator, the expression of $\dot{x}$ will be added to P. Once derived, the obtained expression will be $\ddot{x} = a(-ax^2y + bx)x^2 + 2a(ax^2y - (b+1)x + 1)xy - (b + 1)(ax^2y - (b + 1)x + 1)$. Deriving polynomials from x and y many times may increase computational complexity without major precision improvement. Therefore, choosing a criterion to stop the filling of P_m is necessary. The more elements P_m will contain, the more the qualitative model will be refined, so this choice is the search for a trade between precision and complexity. This criterion must be chosen before the execution, so we still have a problem with non-instantiated systems. If we have no information about the needed precision of the abstraction nor the usefulness of new reference values, the criteria will have to be chosen arbitrarily. In our Brusselator system, with a criterion of a maximum number of derivations of 2, P_1 and P_2 are both initialized to $\{x, y\}$. Then $F[mode_1][x]$ and $F[mode_1][y]$ are added to P_1. $I[mode_1]$ is also added as is the transition condition from $mode_1$ to $mode_2$. At that time, we have $P_1 = \{x, y, \dot{x}, \dot{y}, x - y\}$. Then, each element is derived: as $\dot{x}$ and $\dot{y}$ are already in P_1, they are not added. However, $\frac{d(x-y)}{dt}$, $\ddot{x}$ and $\ddot{y}$ are.

Once P is computed, the next step is to take every $p \in P$ and use a polynomial solver to solve the equation $p = 0$. The obtained solutions, expressed as $X_i = f(X \setminus X_i)$ will give the expression of the nullclines of the state space and will allow a discretization of the variation space of the variable. Each value of X will now be abstracted by comparison to the nullclines.

Definition: We call **qualitative state** of a system the pair (m, qs) with m the current mode and $qs \in \{+, 0, -\}^{|P_m|}$ a vector such that $\forall i \in [\![0, \ |P_m|-1 \ ]\!]$, $qs[i] = -$ if $P_m[i](X) < 0$, 0 if $P_m[i](X) = 0$, and $+$ otherwise. The abstracted values of X will be expressed as a vector of elements from $\{+, -, 0\}$ representing respectively for every $p \in P$ the fact that $p(X)$ is negative, zero, or positive. Therefore, we can define the abstraction function $\alpha : \mathbf{X} \mapsto \{-, 0, +\}^{|X|}$ giving this vector for every value s of X.

The advantage of this method is that it can be applied even on non-instantiated systems: it does not require prior knowledge about the context or the system's objective, meaning it is easy to generalize. However, as we treat systems defined by ODE, there are many chances that the obtained polynomials in P will be neither nilpotent nor idempotent. Therefore, the stopping criteria will be required for the discretization to stop. The problem is that we still need to define these criteria before studying the system and that these criteria will still depend on the case and the specific system.

Moreover, to apply this method, it is necessary to know the explicit formula of the ODE, which means that it does not automatically apply to systems defined by more abstract structures such as bond graphs, causality graphs, or even proportionality relations. Therefore, the method is limited to a subset of CPS where the relations and dynamics are perfectly known with symbolic formulas. We are currently working on generalizing this abstraction to systems defined by causality or proportionality, but we still have a way to go. In the case of explicit equation with non valuated constants, α can be defined

but will not be callable before a complete definition of all the symbolic constants as its return value depends on it.

B. Behavior Exploration

Once the state space's discretization is over and we have defined our abstraction function α, we can now focus on the objective of qualitative reasoning for behavior analysis: the exploration of the hybrid behavior tree. As we do not consider the numerical value of the system's state as necessary, we only seek to propagate the qualitative state of the system and explore all the possible trajectories starting from an initial point. First, we must abstract this departure point to figure out which qualitative state we are in.

Then, while we are not in an absorbent state or in an already explored state, we explore all the neighbors of the current state and add them to the list of qualitative states to be treated. Two structures are used to memorize the qualitative states: a Frontier list and an Explored list. The Frontier keeps the qualitative states from being analyzed, and Explored memorizes the ones already studied. At this point of the computation, we still consider the qualitative states that violate an invariant or that may trigger a discrete transition among the others: the separation will come later. This way, by neighbor propagation, we can compute all the qualitative states where the system can be in its real trace. Adjacent state propagation allows us to determine all possible variation directions from the initial state. To compute the direction of variation, we must compute the Lie derivative: when on a nullcline defined by $p = 0$ separating two states s_1 and s_2, the transition direction will often be unique. To compute it, we use the Lie derivative defined by $L_X(p) = \sum_{X_i in X} \frac{\partial p}{\partial X_i} \frac{\partial X_i}{\partial t}$. If there exist a valuation x of X such that $x \in s_1$ and that the sign of $L_X(p)(x)$ corresponds to the change of sign of p during the transition from s_1 to s_2, then this transition is allowed. Knowing a possible current value of X means that we must have a way to reverse the abstraction function and evaluate X when in the state $(m, s1)$. Once all the possible transitions from the current state s have been found, we add all its successors that have not been visited to the Frontier list and continue until the Frontier is empty. We then add the current state to Explored. As we create states partially based on the sign of the transition conditions, it is essential to monitor which type of qualitative border the trajectories cross. To this extent, each border equation is associated with the type of constraint it was created from. If a nullcline obtained by the zero of a transition condition is crossed, the post-processing of the successor states function detects it, and the qualitative state is replaced by the post-transition state from the target mode in the exploration tree. We note this transition differently in the tree as it is a modal transition, not just a qualitative state change. Once the algorithm stops, we define the automaton based on our execution, where we have the states defined by the pair (mode, qualitative-state). For two states linked by a transition, if $mode_1 = mode_2$, we note the transition as intra-modal. Otherwise, the transition is noted as modal or discrete. We use these qualifications as labels from L.

IV. Introduction of Qualitative Zones

The first direction to improve the efficiency of qualitative reasoning is to process a refinement of the state space partition. This means adding new elements in the set P to compute new borders to compare and situate the numerical values. However, adding more polynomials that correspond to no physical phenomenon does not have great interest. For example, the qualitative states defined with nullclines of 5^{th} orders derivatives will rarely bring useful knowledge for the computation. Therefore, the best option is adding new borders based on equations from previously explored orders. With this idea, we can create a proximity zone around each of the previously computed borders, which will give information about the distance to the coming event when crossing this neighborhood limit. In the beginning, we imagined this as a type of precaution frontier. We studied three main ideas to create such limits that we will now call secondary borders. The first idea we tested was to translate the main borders from a chosen distance d to obtain the neighborhood limit. The advantage of this idea is simplicity: each border can be translated from a specific value in a given direction without requiring heavy computation. However, the choice of the translation direction is entirely arbitrary, and such a frontier will not have any physical meaning, especially in the case of borders defined by nullclines. This is, however, the best solution for the limits defined by $X_i = k$, with X_i a component of X and k a constant in $\mathbb{K}$: the translation direction is then naturally following the axis defined by X_i. The obtained hyperplane is parallel to the main border and defines a proximity area around it at a constant distance d. In the case of nullclines, this solution is not viable.

Another idea was to compute surfaces that are completely parallel to the original nullclines: this would have assured the same distance in every direction and would have visually been very understandable. However, the computation of such surfaces is much trickier than it seems: even in two dimensions, computing the equations defining a curve parallel to another is far from trivial. In two dimensions, in the case of a parametric curve defined at any instant t by $x = f(t)$ and $y = g(t)$ with f and g two functions, the equation of the parallel curves to the parametric curve (x, y) are the parametric curves defined by (x', y') with $x' = f(t) \pm \frac{k\dot{g}(t)}{\sqrt{\dot{f}(t)^2+\dot{g}(t)^2}}$, and $y' = g(t) \mp \frac{k\dot{f}(t)}{\sqrt{\dot{f}(t)^2+\dot{g}(t)^2}}$, with k a constant. This choice would make little sense as we seek simplification and less computation than in numerical analysis. Moreover, such a limit would not have any physical meaning. Consequently, this idea of parallel surfaces does not fit our ambition in the case of nullcline neighborhoods. Finally, the best proposition we developed is using isoclines (i.e., surfaces defined with a function f by $\dot{f} = c \neq 0$. For each element p of P, we must define a value $c_p \in \mathbb{K}$, then use our solver to solve $p = c_p$ and $p = -c_p$. The symbolic results will define the surfaces that will delimit the neighborhood of each main border. Solving these equations does not require specifically

heavy computation, and it corresponds to something concrete: it allows us to know both when a variable from our polynomial elements is about to change its sign when the value of c_p is set very low, but it can also inform us when the same variable is about to vary very fast (and cause an unexpected mistake) when c_p is chosen high. We chose to separate these borders from the previous ones. As we already have our discretization of the state space in qualitative states, we now have another discretization in areas that we call "qualitative zones". If the qualitative states give abstracted knowledge about the distance from the current numerical state to each frontier, the qualitative zones offer an abstraction of the distance separating it from the nearest borders. This allows us to define a coordinate system to locate the CPS in its state space and anticipate and reason about the situation and the coming events.

Definition: We call **Qualitative Position** the triplet $(mode, qualitative\_state, qualitative\_zone)$ of a system. This complementary information creates a qualitative state space map designed to locate a numerical state and reason about its successors. Finally, considering qualitative zones is a good criterion to choose between two qualitative behaviors that could not have been distinguished using qualitative states. For example, in the Brusselator system defined in (1)

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

with a and b two positive constants, the analysis of qualitative states and transitions shows that the system is cyclic around the convergence point. However, using this knowledge to deduce whether the trajectory will be convergent without proceeding to numerical computation is impossible. The consideration of a proximity zone around the nullclines $\dot{x} = 0$ and $\dot{y} = 0$ and around the stable point allows us to determine, using the transition direction, if the system will converge towards $\dot{x} = \dot{y} = 0$, or stay in a critical cycle around it. With a sufficiently small distance d, the study of the surface defined by $||a - conv|| = d$ or by $|\dot{x}| < d \wedge |\dot{y}| < d$ will show if the convergence is possible or not: if the propagation analysis finds it possible with the Lie derivative that the system enters this qualitative zone, then it is convergent. Otherwise, it will deviate to a limit cycle.

V. Computation and Tool Creation

To build our tool, we created two categories of systems corresponding to different situations regarding the knowledge we have access to. These two categories will be represented using the possibilities of object-oriented programming with two different classes. They will correspond respectively to instantiated and non-instantiated systems. As a primary hypothesis, we suppose that in the non-instantiated system, the precise dynamic equations will be available, as will the invariant constraints and transition conditions. In the case of a system on which more knowledge is available (especially about essential values of the variables, the objectives of the system, or the effects of the environment), we use the other category, which concerns the instantiated systems. Here, a new data structure is added to the system, representing all the landmarks that will be considered to study the system and its behavior.

These two categories are differentiated using two classes of objects, the first one named "System", and the second one, inheriting from the first, is named "Instantiated-System". We use object-oriented programming to impose a structure the systems must follow. As we want to unify methods for polynomial solving, symbolic computation, constraint solving, and graph logic, we developed our program in Python to benefit from its numerous libraries. To be able to solve all the equations symbolically, we use the Sympy library. For each mode taken separately, we discretize the associated state space and compute the continuous automaton.

The discretization of the state space is done by placing all the polynomial equations that define the system in the set P and using the functions of symbolic solving provided by Sympy to resolve the equalities $p = 0$. Sympy also gives us the tools to derive the polynomials according to time, to test if the newly obtained formula is a factor of existing polynomials using its polynomial module, and to create the qualitative state based on the comparison of a value to every polynomial. The results of the resolution of the nullcline formula are given as n-uplets with $n = |X|$, in the form $(f_0(X_i), ..., X_i, ...f_{n-1}(X_i))$, that express the equation of the surface of the border.

We then compute the abstraction function: for each numerical value x of X and for each $p \in P_m$, we compute the sign of $p(x)$ to determine on which side of each of the nullclines the current state is. The qualitative state is then represented by an array of -1, 0, and 1, corresponding respectively to $-$, 0, and $+$ but making operations easier to compute.

For example, in our example, let us suppose that the current mode is $mode_1$, that the numerical value of the system is $(1.5, 4.3)$, and that the parameters have value $(a_1, b_1) = (1.2, 1.4)$. If $P_1 = [x, y, x - y, 1 - (b_1 + 1)x + a_1x^2y, b_1x - a_1x^2y]$ as we computed before, the qualitative state of the system is $[+, +, -, +, -]$. Then, from the system's initial state, we computed an algorithm of qualitative state propagation to explore the behavior and all the possible trajectories of the system based on mathematical properties such as the intermediate value theorem, inequality solving, or Lie derivatives.

In order to propagate the current state to compute the behavior tree of the system, we first look at all the qualitative states in contact with each found state. This is done by taking the array of a state as a reference and by creating copies of it where one digit is changed to a possible adjacent state (in the direction of the intermediate value theorem), meaning that a 1 or a -1 can change to 0, and that a 0 can be converted either to a 1 or a -1. In the case of a state $s_1 = [1, 1, -1, 0, -1]$, its computed possible successors will be $[0, 1, -1, 0, -1]$, $[1, 0, -1, 0, -1]$, $[1, 1, 0, 0, -1]$, $[1, 1, -1, 1, -1]$, $[1, 1, -1, -1, -1]$ and $[1, 1, -1, 0, 0]$.

Then we must check whether each of these possible successors exists or not. It means that we must use a constraint solver and check if the conjunction of all the inequalities defining

the states can be $True$ at a point of the space. If it can, the state exists, and we keep it for the next steps. Otherwise, the state does not exist, and we forget it. Moreover, if the new state corresponds to an invariant violation, we keep it in a special category: we will still study the feasibility of the transition, but keep it as an invalid one. The difficulty arising in this step is that Sympy does not allow constraint solving for inequalities: only equality solving is allowed when it comes to a conjunction of equations. Therefore, we used a second Python library allowing symbolic computation specifically for this task, which is Z3. To this extent, we had to design a translation program, converting Sympy expressions to Z3 equations and also converting the variables. Z3 uses an SMT solver to solve a conjunction of symbolic inequalities on a specified set of variables and returns "sat" if the created problem admits a solution. Therefore, to check if a state exists, we convert all the equations and inequalities that characterize it to the Z3 format, create a problem based on the conjunction of all these inequalities, and call the Z3 SMT solver to know whether a solution exists.

Once all the neighbors of a qualitative state s_1 have been isolated, we must then determine which are successors and which are predecessors. As explained, we do this by computing the Lie derivative of the equation corresponding to the border. We execute it with a scalar product between $[\frac{\partial p}{\partial X_i}$ for $X_i \in X]$ and $[\frac{\partial X_i}{\partial t}$ for $X_i \in X]$, with $p = 0$ defining the border. Ideally, this should be evaluated on a numerical value x of X located in the current state in order to determine the sign of this scalar. However, we have not yet found an expression for the concretization function that could give for any qualitative state the corresponding numerical values. To bypass this problem, we use the constraint solver in Z3. We create a virtual state of size $|P_m| + 1$, with the $|P_m|$ first elements being the elements of the initial state, to which we associate a 1 if crossing the border turns p from negative to positive, and -1 otherwise. This new element is linked to the equation obtained with the Lie derivative formula. Then, we call the constraint solver again to determine if this virtual qualitative space exists. If the associated problem is satisfiable, then the Lie derivative on the border allows the transition, and the state s_2 considered as a possible s_1 successor is an actual successor of s_1. We add s_2 to the list of successors of s_1 and s_1 to the list of predecessors of s_2. Otherwise, the transition from s_1 to s_2 is impossible, so we do not add s_2 as a real successor of s_1.

With all the intra-modal transitions computed for each mode, the algorithm now browses the states corresponding to a modal transition constraint crossing. As the natures of the equations defining the qualitative states are kept, one can refer to them and observe which actual successors of s_1 change the value of the associated digit. When one transition to such a state is found, a modal transition to the corresponding target mode replaces the previously computed intra-modal transition. The reset function activates to determine in which qualitative state from the new mode the transition will arrive.

In the end, the algorithm returns an automaton expressed as a multi-dimensional dictionary, where the identifier of each mode and qualitative state are used as keys, giving for each of these states its predecessors and its successors. The state corresponding to the initial value of the system contains the string "start" in its predecessors list.

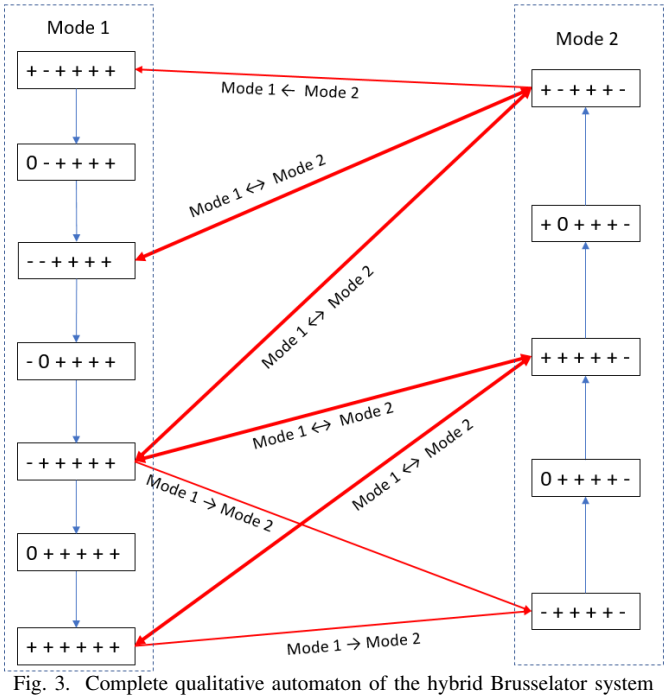

Fig. 3. Complete qualitative automaton of the hybrid Brusselator system

Applied to the hybrid Brusselator and translated into a graphic version, the algorithm gives us the result shown in Figure 3 for the hybrid automaton.

In this figure, the blue vertical arrows correspond to intra-modal (or qualitative) transitions, while the red (oblique) ones are the modal (or discrete) transitions. The thin red arrows represent the one-way transitions, and the thick ones represent transitions that can happen in both directions. No label was added to qualitative transitions, while discrete transitions are labeled with their respective direction. The chosen parameters values in this model were $(a_1, a_2, b_1, b_2) = (1.2, 3.6, 1.4, 2.5)$. The labels of the modes describe the sign of the elements of P_i in the same order. The computed sets P_i are respectively equal to $\{1 - (b_i + 1)x + a_i x^2 y, b_i x - a_i x^2 y, x, y, x - y\}$, containing equations corresponding respectively to the two flow equations, the two invariant constraints and the transition condition. In each state, the set of elements from $\{-, 0, +\}$ characterizes the sign of the corresponding elements from P_i. Many qualitative cycles coexist, all of them transiting by the two modes. There is no behavior staying in only one of them. We now have the complete qualitative automaton of the CPS. Both the discrete transitions and intra-modal behavior are included, and the trajectories are shown more precisely than in traditional hybrid automata.

Now, we must compute the different qualitative zones and their interactions with qualitative states. The distances d_i between every border and its associated secondary limits must be entered as an argument: we did not automatize the choice

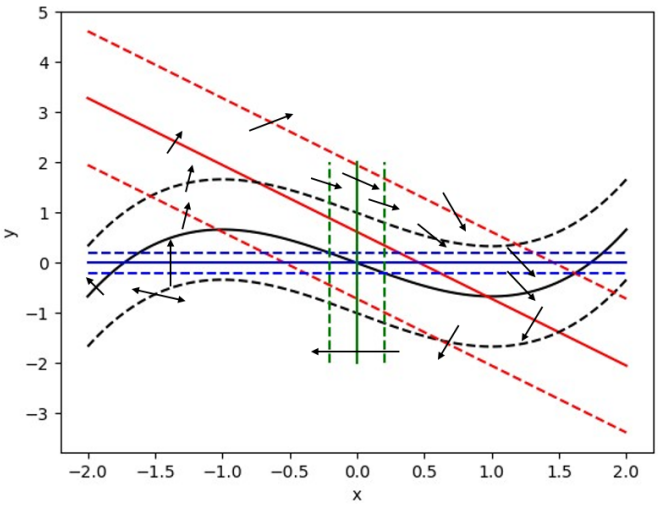

Fig. 4. Qualitative map of a Van der Pool oscillator system

of the value of the isocline yet. Once we have the equations of the different isoclines, we aim to determine how they fit into the qualitative states. This involves a new call on the constraint solver: we can still create virtual states with a new constraint expressed by the equality corresponding to the isocline and make it match different qualitative states in the solver. If the problem is satisfiable, then the qualitative zone defined by the isocline exists in the qualitative state. Knowing which zone exists on each qualitative state and computing once more the transition direction creates a qualitative map that gives much knowledge about the trajectories of the system, with a good balance between qualitative and numerical information. Here, we gave an example of the result we can obtain on a Van der Pol oscillator: this system is a continuous system defined by equation 2, where c is a constant.

$$\begin{cases} \dot{x} = 10(y + x - \frac{x^3}{3}) \\ \dot{y} = c - x - \frac{3y}{4} \end{cases} \tag{2}$$

This system's few equations and borders make it an excellent example, as the qualitative map is still understandable and not too charged. In Figure 4, we drew the main borders with plain lines and the associated secondary borders with dotted lines. This representation corresponds to an instance of the system where c is arbitrarily set to 0.465. The color code associates each border with its proximity limits. For both the straight lines (hyperplanes in two dimensions), we arbitrarily chose a distance of 0.2 to place their proximity limits. We computed the isoclines with $\dot{x} = \pm d_1$ and $\dot{y} = \pm d_2$ where $d_1 = 10$ and $d_2 = 1$. Finally, the black arrows show the transition direction the system allows with the computation of the Lie derivatives. Double arrows highlight that both transition directions are possible at a point of the border.

VI. Application to Piloting and Supervision

We can now automatize the process of creating the qualitative model and its comprehension to generate a hybrid automaton representing the system's behavior. We will now develop some applications of this study of the CPS and the possibilities offered by both the qualitative model and the hybrid automaton. The main application we propose is the creation of an automated pilot to guide numerical simulations of the system or to supervise its real-time execution. As the algorithm computed for each qualitative state all its successors, a pilot using the created structures can anticipate the future transitions and raise an error signal when an unplanned transition happens. This ability to detect miss-events allows correcting errors in the systems simulation, or improving resilience to uncertainty. Moreover, using the qualitative zones, the pilot can not only verify but also prevent the transitions and events in a simulation or a running system, meaning that it can adapt its policy and vigilance to the situation and the expected future events. For example, if a simulation is aimed at measuring precisely the instant of specific events in the execution of the system, the qualitative automaton and the abstraction function can cooperate to precise the simulation among the qualitative spaces and zones. This localization can be used to know if an important transition is about to happen (which means that the simulation step is to high), if one of the variables is about to have a very fast variation (meaning that the step should also be reduced as caution measure), or if, on the contrary, nothing is supposed to happen in particular for the next time steps. The adaptation of the simulation step using the qualitative position of the system has been presented in one of our previous works, and the results have been pretty positive. Our works on automatic adaptation of simulation steps based on qualitative positions were based on the concept of security area and maximum variation. We defined a reference simulation time step d_t that will be used in the absence of specific instruction and an integer k to serve as the numeration base (common values such as 2 or 10 will be preferred to simplify the computation). Using smaller values of k improved precision while using larger favored computation time. The work aimed to adapt the time step d_t to the qualitative position of the system to avoid useless computation in zones without anything special and to focus on zones where transitions, deviations, and errors are the most likely to happen. When in the neighborhood of a transition/event, the pilot imposed a reduction of the time step in order to locate as precisely as possible the exact instant of the event and to avoid rollback [14]. When in the neighborhood of a qualitative transition, d_t was reduced by the chosen factor k to impose more precision on what we call critical areas. Moreover, a precaution threshold must also be defined for each $|\dot{X}_i|$ with $X_i \in X$: a derivative with a high absolute value implies that with a constant time step, the simulation may deviate more from the real trajectory than otherwise. A reduction of d_t must also be implemented to consider this. Therefore, we impose the value of $\max_i(\lceil \log_k(|\dot{X}_i dt|) \rceil)$ to be restricted at each instant. This formula allows us to reason with orders of magnitude [22] and creates a discriminant criterion to fix the numerical steps based on the value and the changes of the derivatives. This method is based on quantized states simulation [23]–[25], and creates a compromise between qualitative and numerical reasoning, joining what Kuipers considered as semi-qualitative

reasoning. Consequently, we can compensate for the effects of any strong variation with an adapted reduction of d_t. The limit value imposed to $\max_i(\lceil \log_k(|\dot{X}_i dt|)\rceil)$ will depend on the researched precision of the model, which is not something automatically quantifiable. Therefore, the intervention of a human agent is required. As a result, we can upgrade the precision of a numerical simulation compared to a constant-step execution and get results with initial time steps that would not have allowed a classic simulation to work. The detection of the precise instant of the transitions is also more precise because adapting the simulation step near the neighborhood limits implies that any post-transition point will be in immediate proximity to the transition border. Consequently, the deviations caused by late detection of the transition threshold decrease. We also noted a reduction of false transitions caused by significant inertia on the variation of the variables when some derivatives have high values. As the pilot automatically reduces the time step in these situations, the inertia is canceled due to the high number of computation points and does not cause any false transition that could completely deviate the simulated trajectory from the actual behavior. Moreover, the execution time of an adaptive simulation is far lower than for a precise simulation with constant and small time steps. The simulator saves time when the system is not in any proximity zone and has no high derivative among its variables. It keeps it for areas of the state space where we seek precision and reliability. Therefore, simulation piloting using qualitative reasoning offers an interesting trade between time complexity and result precision.

In the case of CPS monitoring, it is also possible to change the frequency and the quality of the sensor sampling to optimize the use of resources and keep a good quality of results. By artificially modifying the quantization precision during a simulation, it is also possible to modify the precision of a simulation and to use better-quantized values in more critical zones. However, adding decimals and precision in the values of the system after having decided to suppress the knowledge to improve the computation time is still a problem: earning knowledge to get a finer quantification is not possible without sensors or at the risk of choosing false values. Now that we can nearly automatize the creation of such a qualitative pilot, it is almost possible to generalize it to any kind of CPS expressible with ODE dynamics. We still need to define some parameters, such as a reference time-step that strongly depends on the working context of the system, and a general rule can not choose that for any CPS. We will also have to fully automatize the definition of such a pilot using the qualitative analysis presented above.

VII. Limits and Future Works

The main objective of our contribution was to automatize as much as possible the creation of a qualitative model, its study, and its application to many tasks such as simulation piloting or execution supervising. If our results show that we improved the pre-existing abilities in that area, we still mentioned a few times in this article that we did not find a way to automatize every segment of the process. Many points still require human intervention, such as the choice of the number of times each polynomial function defining the system should be derived at most. Each time a parameter has to be set during the process (also including the distance/curve to apply to the proximity limits to draw the nullclines), the optimal choice of the value requires knowing the context, the application, or the specifications of the system. This is knowledge we cannot access at early stages of development. Therefore, the intervention of a human agent is still necessary to complete the process. To fully automatize it, we thought about taking ideas from naïve physics and common sense reasoning [26] to choose a theoretical best value based on prior conceptions, databases of similar systems, or ontological reasoning about the functional dependencies. However, this is only at the stage of ideas, and we have yet to confront them to experience.

The main difficulty we encounter is the challenge posed by solving equations and inequalities with both symbolic variables and constants. Actually, solving $ax^2y-(b+1)x+1 > 0 \ \wedge \ bx - ax^2y < 0$ with a and b constants is far more difficult when a and b are only symbolic and not valuated. In Python, we have not yet found a way to solve the conjunction of inequalities and to compute the successors of a state in the case where the constants of the system are not or partially valuated because most of the available solvers consider those symbolic constants as other variables and therefore give a solution including a (wrong) value of these parameters. This implies that we are still unable to work on the design space of the system and at very early stages of development. It is an important limitation regarding the importance of qualitative reasoning in CPS design and conception phases. A major progress margin for our work is then to find a solver able to deal with both symbolic variables and constants and to express the existence of a solution of a complex equation system depending on the value of the constants. We thought about using online tools such as wolfram alpha or combining Python with other languages that could offer an adapted solver. Beyond that, we are working on generalizing these advances to more types of systems, defined by more abstract structures such as causality or bond graphs. As causality relations are expressed as influences, causes, and dependencies and not as equations, we must first resolve the previous problem. This objective will expand our production to systems that cannot be expressed with the formalism we used. Moreover, with a few adjustments, it will be possible to extend our tool for systems defined with functions not only polynomial but also rational fractions: as $\mathbb{K}(X) = \mathbb{K}[X] * (\mathbb{K}[X] \setminus \{0\})$, what we exposed to polynomial resolution should apply to rational fractions under the condition of managing cautiously the zeros of the denominators.

We will also improve the capacities of our qualitative models by adding new analyses on the different equations. For example, using works on semi-qualitative reasoning [12], [27], we think that with a better analysis of the flow equations $F_{m,i}$, it will be possible to pilot more complex simulations integrating notions of uncertainty. If we can compute the zones of the

state space where each of the $F_{m,i}$ is a contraction (i.e., where $\exists k \in (0,1)$ such that $F_{m,i}$ is k-Lipschitz), the management of uncertainty will become easier as we will know which areas of the state space can tolerate uncertainties (if a function f is a contraction, an initial uncertainty on f will only reduce with time). This knowledge will give more efficiency to qualitative pilots for simulation and supervision as they can use it to limit artificial uncertainty and lower the simulation steps on non-contraction zones, and give up some precision to save time when all the dynamic equations are contractions.

Finally, by combining qualitative abstraction of CPS with one of our other research axis based on the simplification of functions based on orders of magnitude and orders of growth, we could also include in the abstraction process the systems defined with algebraic differential equations: by simplifying the expression of these equations to piecewise continuous functions based on local properties, we would study these new expressions as we did with polynomials in this article to earn more generality to the tool.

VIII. Conclusion

We have proposed a tool that allows the complete qualitative study of many CPS based on their expression, using elements from different qualitative reasoning methods. We are now able to observe the entire qualitative automaton of the behavior of the system, add a new layer of abstraction with our concept of qualitative zones, and use them to pilot various applications of the system, such as numerical simulation, conception, or real-time monitoring. The process is yet to be fully automatized and generalized to other representations of physical systems to extend its generality and efficiency.

References

[1] M. Mitchell, *Complexity: A guided tour*. Oxford university press, 2009.

[2] T. A. Henzinger, "The theory of hybrid automata," in *Verification of digital and hybrid systems*. Springer, 2000, pp. 265–292.

[3] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition*. MIT Press, 2017.

[4] A. L. Brown, *Qualitative knowledge, causal reasoning, and the localization of failures*. MIT Artificial Intelligence Laboratory, 1974.

[5] J. De Kleer, *Qualitative and Quantitative Knowledge in Classical Mechanics*, ser. AI-TR-. Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1975.

[6] P. J. Hayes, "The naive physics manifesto," *Expert systems in the microelectronic age*, 1979.

[7] B. Smith and R. Casati, "La physique naïve: un essai d'ontologie," *Intellectica*, vol. 17, no. 2, 1993.

[8] P. J. Hayes, *Readings in qualitative reasoning about physical systems*. Morgan Kaufmann, 1985, ch. The second naive physics manifesto, pp. 46–63.

[9] K. D. Forbus, "Qualitative process theory," *Artificial intelligence*, vol. 24, no. 1-3, pp. 85–168, 1984.

[10] B. Kuipers, "Qualitative simulation," *Artificial intelligence*, vol. 29, no. 3, pp. 289–338, 1986.

[11] P. D. Louise Travé-Massuyès, *Modèles et raisonnements qualitatifs*. Hermes, 10 2003.

[12] D. Berleant and B. J. Kuipers, "Qualitative and quantitative simulation: bridging the gap," *Artificial Intelligence*, vol. 95, no. 2, pp. 215–255, 1997.

[13] S. Medimegh, J.-Y. Pierron, and F. Boulanger, "Qualitative simulation of hybrid systems with an application to sysml models." in *MODELSWARD*, 2018, pp. 279–286.

[14] O. Bouissou, A. Chapoutot, and S. Mimram, "Computing flowpipe of nonlinear hybrid systems with numerical methods," *arXiv preprint arXiv:1306.2305*, 2013.

[15] A. Tiwari and G. Khanna, "Series of abstractions for hybrid automata," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2002, pp. 465–478.

[16] Y. Selvaraj, W. Ahrendt, and M. Fabian, "Formal development of safe automated driving using differential dynamic logic," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 988–1000, 2022.

[17] H. Zaatiti, L. Ye, P. Dague, J.-P. Gallois, and L. Travé-Massuyès, "Abstractions refinement for hybrid systems diagnosability analysis," in *Diagnosability, Security and Safety of Hybrid Dynamic and Cyber-Physical Systems*. Springer, 2018, pp. 279–318.

[18] P. J. Mosterman and G. Biswas, "Diagnosis of continuous valued systems in transient operating regions," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 29, no. 6, pp. 554–565, 1999.

[19] O. Maler, "Algorithmic verification of continuous and hybrid systems," *arXiv preprint arXiv:1403.0952*, 2014.

[20] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in *International Hybrid Systems Workshop*. Springer, 1991, pp. 209–229.

[21] O. Maler and G. Batt, "Approximating continuous systems by timed automata," in *International Workshop on Formal Methods in Systems Biology*. Springer, 2008, pp. 77–89.

[22] O. Raiman, "Order of magnitude reasoning," in *Readings in qualitative reasoning about physical systems*. Elsevier, 1990, pp. 318–322.

[23] E. Kofman and S. Junco, "Quantized-state systems: a devs approach for continuous system simulation," *Transactions of The Society for Modeling and Simulation International*, vol. 18, no. 3, pp. 123–132, 2001.

[24] F. E. Cellier, E. Kofman, G. Migoni, and M. Bortolotto, "Quantized state system simulation," *Proc. GCMS'08, Grand Challenges in Modeling and Simulation*, pp. 504–510, 2008.

[25] X. Floros, F. Bergero, F. E. Cellier, and E. Kofman, "Automated simulation of modelica models with qss methods-the discontinuous case," in *8th International Modelica Conference*, 2011, pp. 657–667.

[26] E. Davis and G. Marcus, "Commonsense reasoning and commonsense knowledge in artificial intelligence," *Communications of the ACM*, vol. 58, no. 9, pp. 92–103, 2015.

[27] J. Jerray, "Orbitador: A tool to analyze the stability of periodical dynamical systems." in *ARCH@ ADHS*, 2021, pp. 176–183.