Raisonnement qualitatif et conception de systèmes complexes

Qualitative reasoning and complex systems design

Thèse de doctorat de l'université Paris-Saclay et CEA LIST

École doctorale nº 580, Sciences et technologies de l'information et de la communication, STIC
Spécialité de doctorat : Informatique
Graduate School : Informatique et Sciences du Numérique
Référent : CentraleSupélec

Thèse préparée dans les unités de recherche **LIDEO (Université Paris-Saclay, CEA LIST)** et **LMF (Université Paris-Saclay, CNRS)**, sous la direction de **Frédéric BOULANGER**, professeur à CentraleSupélec, et le co-encadrement de **Jean-Pierre GALLOIS**, ingénieur-chercheur au CEA

Thèse soutenue à Paris-Saclay, le 13 novembre 2024, par

Baptiste GUEUZIEC

Composition du jury

Membres du jury avec voix délibérative

Laurent FRIBOURG Professeur, CNRS	Président
Goran FREHSE Professeur, ENSTA Paris Tech	Rapporteur & Examinateur
Hans VANGHELUWE Professeur, University of Antwerp, Université McGill	Rapporteur & Examinateur
Ada DIACONESCU Maître de conférence, Télécom Paris Tech	Examinatrice
Lina YE Maître de conférence, CentraleSupélec	Examinatrice

Titre : Raisonnement qualitatif et conception de systèmes complexes

Mots clés : Raisonnement qualitatif, Systèmes hybrides, Abstraction, Etats qualitatifs, Espace de design

Résumé : Les systèmes complexes sont une catégorie de systèmes impliquant un nombre important de composants, de variables et de relations entre ceux-ci. L'étude et la conception de tels systèmes constituent des domaines de recherche importants ayant des répercussions sur beaucoup d'autres sujets et disciplines. La plupart de ces systèmes étant dits hybrides et incluant donc simultanément des comportements discrets et continus, leur étude implique une complexité inhérente à la cohabitation de variables de différentes natures. Étudier de manière générale l'ensemble des comportements et aider à la compréhension et à la conception de ces systèmes est donc une tâche ardue. Nous nous sommes intéressés à l'utilisation du raisonnement qualitatif afin de créer, simplifier, et exploiter les discrétisations de l'espace d'états de ces systèmes. Nous avons étudié et testé la pertinence de l'exploitation de ces résultats pour la prévision, le pilotage et l'amélioration de simulations numériques, pour la supervision de l'exécution de systèmes réels ainsi que pour l'assistance à la conception de systèmes et au choix de leurs paramètres.

Title: Qualitative reasoning and complex systems design

Keywords: Qualitative reasoning, Hybrid systems, Abstraction, Qualitative states, Design space

Abstract: Complex systems constitute a category of systems involving a large number of components, variables, and relationships defining their behavior. The study and design of such systems are important research areas with implications for many other subjects and fields. As most of these systems are hybrid, involving both discrete and continuous behaviors, their study must integrate the inherent complexity associated with the cohabitation of variables of different natures. It is, therefore, an arduous task to study all these behaviors in a coherent way and to help understand and design such systems. We were interested in using qualitative reasoning to create, simplify, and exploit the discretization of the state space of these systems. We have studied and tested the relevance of using these results for the prediction, control, and improvement of numerical simulations, for supervising the execution of real systems, and for assisting in system design and parameter selection.

cea

CentraleSupélec

LMF

Laboratoire
Méthodes
Formelles

AKNOWLEDGEMENTS

J'adresse mes remerciements à Frédéric Boulanger, mon directeur de thèse, pour l'encadrement et le regard critique apportés à mon travail, et pour son aide précieuse particulièrement sur les plans technique et rédactionnel. Son expérience et son recul ont été décisifs dans le déroulement et l'orientation de mon travail de recherche.

Merci à Jean-Pierre Gallois, mon encadrant pour ce doctorat effectué au CEA. Ses conseils réguliers et nos nombreuses discussions ont permis à cette thèse de s'orienter vers ce qu'elle est aujourd'hui. Son calme et sa positivité ont su contrebalancer mes inquiétudes et mes doutes lorsque cela a été nécessaire.

Je remercie chaleureusement les professeurs Goran Frehse et Hans Vangheluwe pour avoir accepté de relire et rapporter mon manuscrit et de participer à mon jury de thèse. Je remercie également Pr Laurent Fribourg, Dr Lina Ye et Dr Ada Diaconescu qui ont accepté de prendre part au jury et d'examiner ma soutenance de thèse.

Je remercie l'ensemble de mes collègues du CEA LIST avec qui j'ai passé ces trois années de doctorat. Les nombreux échanges tant professionnels que personnels que nous avons partagés ont été particulièrement enrichissants. Je me permets de remercier particulièrement Frédérique Descréaux pour l'énergie dépensée dans l'adaptation de mes conditions de travail à mes impératifs de santé.

Merci infiniment à mes parents, à ma famille, et à mon entourage qui auront été d'un support indéfectible durant ces trois années parfois difficiles et plus généralement depuis le début de mes études.

Enfin, j'adresse un remerciement tout particulier au Professeur Bertrand Devaux et au Docteur Benoît Crépon pour m'avoir soigné de mon épilepsie et m'avoir permis d'effectuer ce doctorat dans les meilleures conditions de santé possibles.

Contents

List of Symbols

$\mathbb{N}$ the set of natural numbers,

$\mathbb{R}$ the set of real numbers,

$\mathbb{K}$ a given field structure,

$\mathbb{K}^*$ a given field structure without the 0 element,

$\mathbb{K}[X]$ the ring of the polynomials on a variable or a set of variables X with factors in $\mathbb{K}$,

$\mathbb{K}(X) = \mathbb{K}[X] * (\mathbb{K}[X] \setminus \{0\})$ the ring of the rational fractions on a variable or a set of variables X with factors in $\mathbb{K}$,

$\mathcal{C}^n$ the set of continuous functions that can be differentiated n times and whose n^{th} derivative is continuous,

$\mathcal{E}$ the set of functions expressible as a sum, product, or combination of exponential and logarithmic functions,

$\mathbf{X}$ the variation space (domain) of a variable X.

1 - INTRODUCTION

Complex systems [1] are defined by a large number of components, variables, and interactions defining their behavior. The more interactions, parameters, variables, and feedback relations these systems have, the more difficult it is to model them and the more complex these models will be to produce and use. As even the best models of the simplest systems do not allow a perfect representation of reality [2], the situation is even less satisfactory when facing greater complexity.

Complex systems are present in many disciplines, such as mechanics, sociology, biology, chemistry, electronics, and computer engineering. Such systems are the norm in modern societies, which face critical challenges that require increasingly elaborate solutions.

Among the long list of complex systems that could be modeled with mathematical and computational tools, the cyber-physical systems (CPS) category is of great importance, considering modern stakes. CPSs [3] have the singularity of being composed of a classical physical part, controlled and managed by a computational and software entity. Such a control structure implies the existence of various control modes managed by the software component of the system. When in finite number, these modes can be symbolized with a discrete variable whose value will influence the whole system's behavior. Therefore, the system exhibits discrete and continuous variables, further complicating its analysis and modeling. Such systems are called hybrid and are increasingly present in computational engineering [4] as software components control more systems. While systems are increasingly connected and CPSs are omnipresent in modern industries and societies, the stakes related to these systems become more urgent and essential. Particularly, ensuring an acceptable level of reliability and conformance of any commonly used system is paramount, as anomalies and misconceptions could have disastrous consequences. To face these challenges, the CEA LIST uses and develops modeling platforms such as Papyrus and seeks more behavioral modeling paradigms like qualitative modeling.

To represent hybrid systems, the formalism of hybrid automaton [5, 6, 7] is frequently chosen. This structure is widely used to represent various hybrid systems of different natures, uses, or complexity and allows an interesting range of reasoning, computations, and operations on the represented systems. It is generally accepted to be a reliable tool for highlighting the critical properties of the models.

However, a major problem with this representation is that it only gives a theoretical expression of the system's behavior and does not allow any formal study or in-depth analysis of the possible traces of the system, as the state space is not profoundly considered in the model. Therefore, the behavioral analysis of many complex systems requires numerical simulations to test and observe the theoretical trajectories of a system, its potential errors and failures, and its anticipated behaviors. As the computation of such simulation can be very costly for the most complex systems, avoiding complexity in the representation and study of systems when possible is a significant stake. Less complexity would allow us to focus on the behavioral essence of a system and on its defining characteristics to deduce as much information and knowledge as possible without implying unnecessary computations and data.

1.1 . Contribution

This thesis's contribution consists of improving and formalizing the qualitative modeling process that can be applied to CPSs represented by rational ordinary differential equations. Our method will allow us to abstract the system's state space in a discrete set of qualitative states, which is more suitable than classic numerical models for making symbolic computing, proof, and testing. From this abstraction, we will present a method to compute the qualitative behavior of systems using a new structure of qualitative automaton strongly inspired by hybrid automata. The major improvement we bring to the qualitative abstraction process is introducing the concept of qualitative zones that expands the possibilities of qualitative reasoning by increasing the quantity and quality of knowledge contained in the models. We developed a prototype tool aiming to automatize the abstraction process and create the qualitative automaton using various tools such as symbolic computing or SMT solvers. This method should further allow for diverse analyses and upgrades in the study of CPSs at different points in their development and life cycle. We also provide reflections on possible use cases and preliminary results from applying our models in such situations.

1.2 . Thesis Outline

The upcoming document is structured as follows:

- chapter 1 introduces the subject of this thesis and the stakes associated with it,
- chapter 2 presents the scientific context surrounding the thesis subject. It develops the central notions and literature related to CPSs, system modeling, hybrid automata, numerical simulation, symbolic computing, common sense reasoning, and qualitative abstractions,
- chapter 3 presents a framework for CPS modeling, design space exploration, and qualitative reasoning and simulation techniques,

- chapter 4 develops a system abstraction method using qualitative reasoning and the creation of a new qualitative automaton structure based on the analysis of systems equations.
- chapter 5 introduces our main contribution to improving qualitative reasoning with the concept of qualitative zones,
- chapter 6 develops our implementation choices to create a prototype tool that can execute the described abstraction process,
- chapter 7 presents different potential use cases where the developed qualitative reasoning methods could be applied to improve the performance or capabilities of existing tools,
- chapter 8 questions the generalizability of the presented process and suggests avenues and results that could lead to wider use of qualitative reasoning on more complex or less convenient CPSs,
- chapter 9 positions our contributions among the different works proposed in related fields,
- chapter 10 concludes this thesis, reviews our contributions, and presents perspectives to widen the possibilities and fields for future research.

1.3 . Résumé en Français

Les systèmes complexes [1] sont des systèmes caractérisés par un grand nombre de composants, de variables et d'interactions régissant leur comportement. Plus ces systèmes impliquent d'interactions, de paramètres, de variables et de relations de rétroaction, plus il est difficile de les modéliser et plus ces modèles seront complexes à produire et à utiliser. Puisque même les modèles les plus complets des systèmes les plus simples ne permettent pas une représentation parfaite de la réalité [2], la situation est encore plus délicate lorsque l'on est confronté à une plus grande complexité. Les systèmes complexes sont présents dans de nombreuses disciplines, telles que la mécanique, la sociologie, la biologie, la chimie, l'électronique et l'informatique. Ces systèmes sont devenus primordiaux dans une société moderne confrontée à des défis de plus en plus critiques et dont les problèmes nécessitent des solutions d'autant plus complexes. Parmi la longue liste des systèmes complexes susceptibles d'être modélisés à l'aide d'outils mathématiques et informatiques, la catégorie des systèmes cyber-physiques (CPS) revêt une grande importance au regard des enjeux actuels. Les CPSs [3] ont la particularité d'être composés d'une partie physique, contrôlée et gérée par une entité logicielle. La présence d'une telle structure de contrôle implique l'existence de divers modes de fonctionnement dans le comportement du système, dont la commutation est gérée par sa composante logicielle. Lorsqu'ils sont en nombre fini, ces modes peuvent être représentés par une variable discrète dont la valeur influencera le comportement de l'ensemble du système. Ces systèmes présentent donc des variables discrètes et continues, ce qui complexifie davantage leur analyse et leur modélisation. De tels systèmes sont appelés hybrides et sont de plus en plus présents dans l'ingénierie informatique [4] car le contrôle de systèmes physiques par des logiciels est devenu très commun. Alors que les systèmes sont de plus en plus connectés et que les CPS tendent à devenir omniprésents dans les industries et les sociétés modernes, les enjeux entourant ces systèmes deviennent de plus en plus critiques et importants. En particulier, il est crucial d'assurer un certain niveau de fiabilité et de conformité de tout système couramment utilisé, dont les anomalies et les erreurs peuvent avoir des conséquences désastreuses. Pour faire face à ces défis, le CEA LIST utilise et développe des plateformes de modélisation telles que Papyrus et étudie des paradigmes de modélisation permettant de meilleures analyses comportementales tels que la modélisation qualitative. Pour représenter les systèmes hybrides, le formalisme des automates hybrides [5, 6, 7] est fréquemment employé. Cette structure est largement utilisée pour représenter divers systèmes hybrides de nature, d'utilisation ou de complexité différentes et permet un large éventail de raisonnements, de calculs et d'opérations sur les systèmes représentés. Celle-ci est généralement considérée comme suffisamment fiable pour mettre en évidence les propriétés critiques des modèles. Cependant, un problème majeur de ce formalisme est qu'il ne donne qu'une expression théorique du comportement du système et ne permet pas une étude formelle et une analyse en profondeur de ses possibles traces puisque l'espace d'états n'est pas pris en compte de manière approfondie dans le modèle.

Par conséquent, l'analyse comportementale de nombreux systèmes complexes nécessite des simulations numériques pour tester et observer leurs trajectoires, les erreurs potentielles et les défaillances dans les données d'entrée, ainsi que pour analyser les comportements anticipés. Comme le calcul de ces simulations peut être très coûteux pour les systèmes les plus complexes, minimiser la complexité dans la représentation et l'étude des systèmes est un enjeu majeur. Une complexité moindre nous permettrait de nous concentrer sur l'essence d'un système et sur ses caractéristiques principales afin de déduire autant d'informations et de connaissances que possible sans impliquer de calculs et de données superflus.

1.3.1 . Contributions

Les contributions de cette thèse consistent à améliorer et à formaliser le processus de modélisation qualitative qui peut être appliqué aux systèmes cyber-physiques représentés par des équations différentielles ordinaires rationnelles. Notre méthode nous permettra d'abstraire l'espace d'états du système en un ensemble discret d'états qualitatifs, qui seront plus appropriés pour faire du calcul symbolique, de la preuve, et des tests que les modèles numériques classiques. À partir de cette abstraction, nous présenterons une méthode pour calculer le comportement qualitatif des systèmes en utilisant une nouvelle structure d'automates qualitatifs fortement inspirée des automates hybrides. La principale amélioration que nous avons apportée au processus d'abstraction qualitative est l'introduction du concept de zones qualitatives qui élargit les possibilités de raisonnement qualitatif en augmentant la quantité et la qualité des connaissances contenues dans les modèles. Nous avons développé un prototype d'outil visant à automatiser l'exécution du processus d'abstraction et à créer l'automate qualitatif d'un système à l'aide de divers outils tels que le calcul symbolique ou les solveurs SMT. Cette méthode devrait permettre diverses analyses et mises à jour dans l'étude des systèmes cyber-physiques à différents stades de leur développement et de leur cycle de vie. Nous partageons également des réflexions sur des cas d'usage possibles et des résultats préliminaires de l'application de nos modèles dans de telles situations.

1.3.2 . Structure de la thèse

Ce document est organisé selon la structure suivante :

- Le chapitre 1 introduit le sujet de la thèse et les enjeux associés.
- Le chapitre 2 présente le contexte scientifique entourant le sujet de cette thèse. Il développe les notions centrales et la bibliographie relatives aux systèmes cyber-physiques, à la modélisation des systèmes, aux automates hybrides, à la simulation numérique, au calcul symbolique, au raisonnement de sens commun et aux abstractions qualitatives.
- Le chapitre 3 présente un cadre pour la représentation des systèmes cyber-physiques, l'exploration de l'espace de conception, la modélisation qualitative et les techniques de simulation.
- Le chapitre 4 développe une méthode d'abstraction de système basée sur le raisonnement qualitatif et détaille la création d'une nouvelle structure d'automates qualitatifs reposant sur l'analyse des équations des systèmes.

- Le chapitre 5 présente notre principale contribution à l'amélioration du raisonnement qualitatif avec le concept de zones qualitatives.
- Le chapitre 6 développe nos choix d'implémentation pour créer le prototype d'un outil permettant d'exécuter le processus d'abstraction décrit.
- Le chapitre 7 présente différents cas d'usage possibles auxquels les techniques de raisonnement qualitatif développées pourraient être appliquées pour améliorer les performances ou les possibilités des méthodes existantes.
- Le chapitre 8 pose la question de la généralisation du processus détaillé et donne quelques indications et résultats qui pourraient conduire à des utilisations plus larges du raisonnement qualitatif sur des systèmes cyber-physiques plus complexes ou moins triviaux.
- Le chapitre 9 positionne nos contributions parmi les différents travaux proposés dans des domaines connexes.
- Enfin, le chapitre 10 conclut cette thèse, passe en revue nos contributions et présente des perspectives pour élargir les possibilités et les domaines des travaux futurs.

2 - SCIENTIFIC CONTEXT

This chapter introduces the scientific context related to the PhD work. Section 2.1 introduces the cyber-physical systems and the notion of hybrid systems. Section 2.2 summarizes current paradigms and tools used and applied to study and design cyber-physical systems. Section 2.3 details numerical simulation methods mainly used in many domains, notably in system simulation and verification. In addition, section 2.4 develops the notion of computer algebra, which is necessary to deal with partially designed systems. We present in section 2.5 some notions of constraint solving that are helpful in problem-solving. Finally, section 2.6 presents some concepts of common sense reasoning that will be used later in this thesis to abstract systems.

2.1 . Cyber-Physical Systems

Cyber-physical Systems (CPS) [3, 8] constitute a category of systems in which a computer-based monitor controls physical mechanisms. The concept of CPS implies interactions between computation, control, and communication in a physical environment. CPSs require input data and return output results and are supposed to be flexible and adaptable for many different use configurations. CPSs are vastly used in many scientific areas to represent physical, electrical, mechanical, or chemical systems controlled by a discrete process. They involve various fields and approaches such as control and process theories or optimization and design methods [9]. Among modern innovations, autonomous vehicles, military drones, surveillance systems, thermostats, and space vehicles are significant examples of classic CPS. CPSs enable interactions between the computational element of the physical part, the first sending instructions to the second as it receives feedback and data from it. Therefore, CPSs are transforming our interactions with the physical world and our approach to many technical challenges [10]. The computational power and the current interest in CPS come from the increasing precision and reliability of sensors, allowing the delivery of very precise information to the computer pilot in a very short delay. This improved communication between the components allows the use of CPS in real-time applications or more complex or critical systems. Applying these technologies at the scale of a nuclear power plant or even at the scale of a city using the concept of smart cities seems possible today. United with the rise of generative artificial intelligence algorithms, CPS activities will continue to increase.

However, the complexity in such targeted systems is a supplementary stake that must not be underestimated [1]. The notion of complex systems

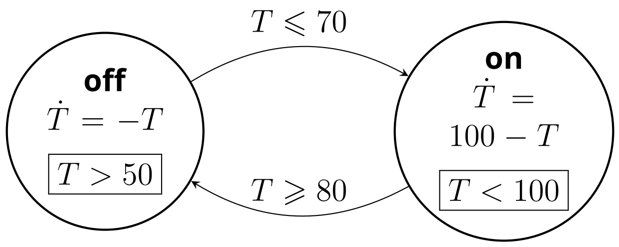

Figure 2.1: Hybrid model of a thermostat

occurs in systems whose physical part is composed of many components interacting with each other. Mitchell compares the challenge posed by high complexities to the difficulty of controlling or predicting the movements of each ant in an anthill.

The presence of many components implies a large number of variables and relations between them. Such complex systems pose major problems at different stages of their lifetime, from the design to the diagnosis or the monitoring. The choice of the parameters, the computational tests, and the monitoring are significant stakes worth improving and simplifying.

By definition, a computer monitor's intervention in a CPS's behavior implies that these systems are hybrid.

In this document, we call *Hybrid System* a system that includes discrete and continuous behaviors and variables.

The hybrid nature of a system depends on the behavior of its variables [11]. Therefore, it is necessary to specify the notion of a variable in a system.

A system includes *variables*, which are measurable quantities with a physical unit, a value, and a variation set (domain).

By definition, the value of a variable can evolve during the considered time period. However, its physical unit and domain are fixed. The physical unit of a variable defines its nature (is it a mass, a distance, energy, ...), and its value is expressed using units from the international system (meter, second, kilogram, ...).

Example 1 (Thermostat) A simple example of a hybrid system is a Thermostat, which produces heat when the environment temperature T_{env} is under a certain threshold and is off when T_{env} is above this threshold. Here, the continuous variable of the system is T_{env} evolving in $\mathbb{R}^+$ and expressed in Kelvins, and the discrete one is $status$, taking either the value on or off and having no corresponding unit.

The heterogeneous nature of these systems implies that the usual modeling, simulation, and reasoning tasks will become more challenging to achieve due to the mutual impacts of the different variables and trajectories of the systems. This supplementary complexity can be suppressed by converting them to fully discrete or continuous systems [12]. However, the loss of knowledge and precision induced by this simplification may be too important in the case of systems exhibiting very different behaviors depending on the value of the discrete variable.

We call *operating mode* (or just *mode*) of a hybrid system a valuation of its discrete variables. Each mode corresponds to a different behavior for the continuous variables and, therefore, a specific operating procedure for the continuous part of the system.

In some references such as [13], hybrid systems are represented using action systems to put more emphasis on the discrete behavior and to highlight the requirements and consequences of the discrete transitions, while the continuous part of the system is associated with the environment of the system. In this approach, the system consists in a "parallel composition between a controller and the environment", noted *controller||environment*.

2.2 . Modeling and Representations

System modeling is an approach aiming to represent, manipulate, and understand complex systems. The model of a system is a mathematical abstraction representing the system. In our context, system modeling is the process of formulating mathematical equations that describe the dynamic behavior of a system. It serves the purpose of understanding, analyzing, and predicting the system's behavior over time [14].

Creating a model requires the definition of the system S to be modeled and of a modeling language. The modeling action on S using the chosen language generates a model M.

M is supposed to contain specific information about S, and the choice of the modeling language will strongly depend on the knowledge to be preserved. System modeling requires making strategic choices about the elements to highlight, as no model can include all the information of S [2].

Systems to model can have different natures. CPSs are, by definition, hybrid as their cyber parts generate mode changes depending on the evolution of their continuous variables. However, the difficulties implied by modeling these systems require first defining modeling concepts and structures for continuous systems and generalizing them to hybrid ones.

2.2.1 . Continuous systems

This section introduces the specific categories of continuous systems [15]. First, it gives some definitions of continuity in a mathematical sense.

Definition 1 (Continuous Function) Let (A, d) and (B, d') two metric spaces and $f : A \to B$ a function. f is continuous at point $a \in A$ if $\forall \epsilon > 0, \exists \delta > 0$

such that $\forall\, x \in A, d(x,a) < \delta \implies d'(f(x), f(a)) < \epsilon$. f is continuous on A if $\forall\, a \in A,\ f$ is continuous at a.

A function f continuous on its definition space belongs to $\mathcal{C}^0$. If derivable and its derivative is continuous on its definition space, then f is $\mathcal{C}^1$.

Definition 2 (Continuous System) Let S be a cyber-physical system with n continuous variables and k discrete operating modes. If $k = 1$, S is said to be a continuous system as no change of mode can occur in the behavior of the system.

Continuous systems contain components characterized by variables taking values in a specific uncountable set (often $\mathbb{R}$) and varying with time. The variation of such variables is expressed using mathematical tools such as differential equations. These systems are characterized at each instant by the value of their variables at this instant, which can either be observed precisely or with uncertainties.

Example 1 *The Brusselator system, presented in [16], is a chemical system described by two variables x and y representing the respective concentrations of two chemical products and whose dynamics is given by the ordinary differential system 2.1:*

$$\begin{cases} \dot{x} = 1 - (b+1)x + ax^2y \\ \dot{y} = bx - ax^2y \end{cases} \tag{2.1}$$

with a and b two positive constants. As this system only has one operating mode and, therefore, does not exhibit any guard condition or discrete transition, it is a continuous system.

2.2.2 . Discrete systems

Discrete systems, or discrete event systems [17, 18], correspond to systems with a state space that is naturally discrete and finite. In this situation, one cannot represent the continuous evolution of the system with the value of the system variables at each time step but by representing the discrete instant at which discrete transitions between the different operating modes occur. These discrete transitions are called events as they correspond to discrete instants on a continuous time scale.

Definition 3 (Discrete Event System) Let S be a cyber-physical system with n continuous variables and k operating modes. If $n = 0$, S is said to be a discrete event system (DES).

In discrete event systems, the mode does not have a meaning as there are no continuous dynamics to influence. Rather than continuous evolution, DESs only exhibit the temporal relations between the different modes and the beginning and end of all system positions.

Example 2 *Let us consider a system made of a house lamp which turns off or on depending on the detected luminosity. Considering coarsely that the lamp has only two possible light power values $x = 0$ and $x = L_{max}$, and that the switch between the two modes is instantaneous (which is a rough but classic abstraction), the system can be modeled using two operating modes, each one imposing a fixed light power x either at 0 or L_{max}, and switching from one to another depending on the relative environment light to a threshold value s_1.*

2.2.3 . Hybrid systems

As already explained, CPS is a specific type of hybrid system. Therefore, we give here the definition of hybrid systems that include the CPS in its range.

Definition 4 (Hybrid System) A hybrid system is a tuple $S = \langle Q, X, I, F, T, P\rangle$ where:

Q	is the current mode of the system (discrete variable), with value q,
X	is a set of n continuous variables X_i with values x_i,
I	is a function mapping each mode to its invariant conditions,
F	is a function mapping each mode to its flow conditions,
T	is a set of discrete transitions, each described by: • their starting mode and guard condition, • their target mode, and reset function
P	is a set of m parameters

Q takes its values on $\mathbf{Q}$ a finite set of modes. The X_i take their values on $\mathbb{K}$ a continuous field where the usual operators $(+, -, *, /)$ and $(<, =)$ are defined. In general, $\mathbb{K} = \mathbb{R}$. The parameters P_i of P are valuated on $\mathbf{P}$.

The invariant conditions of a model represent the structural constraints of the system [19] using semi-algebraic predicates that encapsulate the limits of its authorized behavior.

In some references such as [20], hybrid systems are also composed of algorithms encapsulating the evolution of the environment (both local and global).

Definition 5 (semi-algebraic set) A subset E of the field $\mathbb{K}$ is semi-algebraic if it can be expressed as a finite union of subspaces defined by polynomial equalities and inequalities of the form $P(x_1, ..., x_n)\ op\ 0$ for $(x_1, ..x_n) \in \mathbb{K}^n$ and $op \in \{<, =, >\}$.

The flow, invariant, and guard transitions define semi-algebraic sets in the system state space, allowing further analysis and discretization of the model.

The study of systems of different natures implies different stakes and challenges depending on the aim of the study. The logic analysis of a system will not require the same tools as a temporal study. In the first case, we are interested in the order of appearance of the events of the behavior, while the second also gives importance to the timing of these events.

Set-based or Relational-based notation

In the presented definition of hybrid systems, guard, invariant, and jump conditions can have different representations. These conditions can be represented as sets or as predicates.

Set-based notation implies representing the conditions as the inclusion of a variable in a mathematical set: in the example of the thermostat presented above, the different condition would be expressed as $x \in [60, 70]$ or $x \in [70, 80]$.

On the contrary, relational-based notation expresses these conditions as predicates on the variables: the system presented in Example 1 uses this notation.

Depending on the situation, the chosen notation might differ: the set notation is more adapted when the condition is on a set that cannot be expressed with comparison operators or on a purely symbolic set. On the other hand, when the condition can be more easily expressed using binary operators, relational notation will be preferred.

Hypothesis 1 *We assume that the guard conditions are both necessary and sufficient to provoke the associated transition. Considering the guard conditions as necessary but insufficient would add a stochastic component in the system that would complexify its study. For the same reason, we assume the transition occurs precisely as soon as the guard condition is verified. This hypothesis is not realistic but is an important theoretical simplification.*

Hypothesis 2 *We assume that guard conditions of different transitions from the same initial mode are disjointed:* $\forall\, m_0 \in \mathbf{Q}, \forall\, (\tau_1, \tau_2) \in T[m_0]^2, \tau_1 \neq \tau_2 \implies guard(\tau_1) \cap guard(\tau_2) = \varnothing$.

Definition 6 (State of a system) Let S be a hybrid system according to definition 4. Let $x = (x_i)_{i \in [\![1,\, n]\!]}$ be a valuation of X and $m \in \mathbf{Q}$ the mode of the system at a time t. (m, x) is the state of the system at time t. m is the discrete component of the state and x is its continuous component.

In the case of continuous (resp. discrete) systems, the state of a system will be represented only with x (resp. m, meaning that the state replaces the useless notion of mode).

Definition 7 (State Space of a System) Let S be a hybrid system according to definition 4. The Cartesian product $\mathbf{Q} \times \mathbb{K}$ defines the state space of S.

In the case of a continuous (resp. discrete) system S, the state space of S will be $\mathbb{K}$ (resp. $\mathbf{Q}$).

2.2.4 . Hybrid automata

This section presents the formal representation of hybrid automata as developed in [21]. First, we present a general definition of the concept of hybrid automaton before defining some particular cases.

The hybrid automaton structure is widely used in system modeling [22].

Definition 8 (Hybrid Automaton) A hybrid automaton is a tuple

$$H = \langle Q, X,\ V,\ E,\ Init,\ I,\ F,\ J,\ L\rangle$$

with:

- Q, X, I, and F corresponding respectively to the discrete variable, the set of continuous variables, the invariant constraints, and the flow equations as defined in definition 4 for the associated cyber-physical system. V are the vertices of H representing all its control modes and E its edges between the vertices corresponding to modal transitions.

- $Init$ the set of initial conditions, with predicates constraining the initial values q_0, x_0 of Q and X.

- J a jump condition function that associates to each transition $e \in E$ a predicate. It corresponds to the guard condition of T in definition 4.

- L 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, its specificity, or its reset condition.

Automata are associated with a graphical representation. Therefore, representing hybrid systems as hybrid automata allows us to visually represent their behavior using their vertices as nodes and their edges as arcs linking them. This representation highlights the different trajectories between the system modes by representing the expected transitions between the various operating modes.

Each node can exhibit the flow equations of the system variables in the corresponding mode to show the expected evolution of these variables in this mode and improve the clarity of the visual representation.

Example 3 *For example, we can represent here a thermostat system as a hybrid automaton with:*

- $\mathbf{Q} = \{\textit{on}, \textit{off}\}$

- X *representing the temperature of the system and* $\mathbf{X} = \mathbb{R}^+$

- $V = \{v_1 = \textit{on}, v_2 = \textit{off}\}$

- $E = \{T_1 = \textit{on} \to \textit{off}, T_2 = \textit{off} \to \textit{on}\}$
- $Init = (mode = \textit{off}, T \in [70, 80])$
- $I = \{T > 50, T < 100\}$
- $F = \{\textit{on} : \dot{T} = 100 - T; \textit{off} : \dot{T} = -T\}$
- $J = \{T_1 : T < 70; T_2 : T > 80\}$
- $L = \{(\textit{switch on}, \textit{switch off}, f : T_1 \mapsto \textit{switch off}, T_2 \mapsto \textit{switch on})$

As hybrid automata are here used to represent hybrid systems visually, the same hypothesis about determinism holds. In the general case, we suppose that both jump conditions, dynamics, and reset functions are deterministic to allow abstraction and manipulations that would not fit with stochastic elements. Non-deterministic systems will be treated separately with specific representations and methods.

A hybrid automaton is a classic representation formalism for many systems (for physical, chemical, social, or cyber-physical), and many applications and tools were developed to work specifically on it, such as *HyTech* [23].

As hybrid automata are a very versatile structure, there are many variants and particular versions of this object. The following paragraph will present some specific versions of automata developed in [5].

2.2.5 . Rectangular automata

The first specific category of hybrid automata to be introduced here is the family of rectangular automata [24]. A hybrid automaton is said to be *rectangular* if the flow conditions and variables are independent of the modes and if the continuous variables are pairwise independent [21]. The flow condition defines an authorized set for the derivative of each continuous value X_i of the system, and that set does not change between the different modes.

Definition 9 (Rectangular Automaton) Given that a n-dimensional rectangle is the cartesian product of n closed intervals $[a, b]$ with $a < b$, a hybrid automaton is rectangular if:

- $\forall X_i \in X,\ init(X_i)$ is a singleton
- I has the form of a n-dimensional rectangle
- $\forall X_i \in X,\ F(X_i)$ is a rectangle and does not depend on any $X_j, j \neq i$.
- for each modal transition of E, the associated jump condition j of J has the form $X \in I$ with I a n-dimensional rectangle. $\forall X_i \in X$, if X_i is reinitialized by j, then its reset condition has the form $X_i \in I_i$ with I_i a singleton.

If the flow rectangle is a singleton, the rectangular automaton H is said to be *singular*.

A rectangular automaton expresses a certain level of uncertainty on the flow condition (a rectangular condition is less precise than an equality but can still express a tendency) and highlights the independence between the continuous variables.

An example an a rectangular automaton is given in [25], with the automaton of a robot described by $Q = \{m\}$ with $\mathbf{Q} = \{m_0, m_1\}, X = (x, y, z)$ with $\mathbf{X} = \mathbb{R}^3_+, V = \{v_1 = m_0, v_1 = m_1\}, E = \{T_1 = m_0 \to m_1, T_2 = m_1 \to m_0\}, Init = \{x = y = z = 0\}, I = \{X \in \mathbb{R}^2_+ * [0, 2]\}, J = \{m_0 : X \in [1, 2] * [1, 2] * \{1\}, m_1 : X \in [1, 2] * [-2, -1] * \{1\}\}, L = \{T_1 \mapsto (z := 0), T_2 \mapsto (Z := 0)\}$.

2.2.6 . Variations of rectangular automata

Rectangular automata represent a particular situation that may rarely correspond to concrete systems. The restrictive hypotheses limit the concrete use of such automata in concrete case studies. This paragraph introduces definitions of other types of hybrid automata that are variations from the definition of rectangular automata. This allows for more generalization and a complete view of the existing families of hybrid automata.

Definition 10 (Multi-rectangular Automata) An automaton meeting all the requirements of definition 9 except the equality of the flow conditions between the different modes is *multi-rectangular*.

A multi-rectangular automaton can be considered a piece-wise rectangle automaton, as it locally exhibits rectangular automaton properties of every mode but not on its global structure.

Definition 11 (Triangular Automaton) Given that a triangle is an intersection between a n-dimensional rectangle with a non-zero number m of half-spaces of $\mathbb{R}^n$ defined by an inequality $x_i < x_j$ with $i, j \in [\![1, n]\!]$, a hybrid automaton is a *triangular* automaton if it meets all the requirements of definition 9, with triangular conditions instead of rectangles.

Compared to rectangular automata, the triangular variation represents specific cases where the system variables are not decoupled anymore. In triangular automata, some conditions depend on the value of various state variables. This allows the representation of more complex systems and automata where the value of a variable may influence the others.

Henzinger gives in [5] an example of a triangular automaton with a controller system with three modes and two state variables z and u, whose invariants and guard conditions are expressed using the triangular condition $z \leq u$.

It is to be noted that the reachability problem is decidable in the case of rectangular automata but note for triangular ones as the consideration of triangles instead of rectangles reduces the reliability of knowledge about numerical values [26].

Finally, linear automata [27] also are a special type of automaton generalizing both rectangular, multi-rectangular, and triangular automata. In linear automaton, the constraint sets are defined by any linear equality of inequality (such as $\dot{X} = AX$ with A a matrix in $\mathbb{K}$). This implies that the sets of the defined system can be in the form of any convex polyhedron.

Definition 12 (Linear Hybrid Automaton) A hybrid automaton H is a linear automaton if

- $\forall m \in \mathbf{Q}$, the predicates included in $F(m)$ are convex linear predicates involving only elements from $\dot{X}$.
- $\forall m \in \mathbf{Q}$, the predicates of $I(m)$ and $Inv(m)$ are convex linear predicates over $\mathbb{K}^n$.

From the definition of linear automata, it is also possible to immediately define multi-affine automata [28] by replacing the linear flow predicates with multi-affine functions.

In this thesis, we mainly worked with polynomial automata [29]. Polynomial automata correspond to a family of automata where the different sets are represented using polynomial constraints.

Definition 13 (Polynomial Automaton) Given the field $\mathbb{K}$ on which the variables of X are defined and $\mathbb{K}[X]$ the ring of polynomials on the variables X, a hybrid automaton H is polynomial if each of its sets and conditions is defined using equations and inequalities on $\mathbb{K}[X]$.

Polynomial automata were chosen for their generality as polynomial functions can have various forms of behavior and locally approximate many different types of functions. They are more general and permissive than linear automata.

Remark 1 *Linear automata are specific instances of polynomial automata, only allowing first-order polynomials.*

Rectangular automata and their variations are mainly used as tools to reason about the system's safety and to prove the non-reachability of critical states. As the knowledge about the concrete system behavior is very fuzzy and imprecise, techniques such as formal proof and bisimulation must consider the worst-case scenario among the authorized flow, which demonstrates the reliability of a system under its normal operating conditions.

2.2.7 . Timed automata

We assume, in general, that the behaviors of CPSs are an evolution of their situation according to time. In general, time measurement is considered exogenous to the system as time goes on independently from the system execution. In this situation, time is only a parameter of the system used to evaluate or derive the system's state variables.

However, a category of automaton named timed automata exists.

Definition 14 A timed automaton is a hybrid automaton verifying the definition 8 and such that:

- $\mathbf{X} = \mathbb{R}^n$ with $n = |X|$ and the elements of X are called $clocks$,
- $\forall m \in \mathbf{Q}, \forall x \in X, F(m, x) = \frac{dx}{dt} = 1$,
- $S_0 = (Q_0, \{0\}^n)$, meaning that each clock variable is initialized at zero.

It is to be noted that the flow conditions of timed automata are also independent of the mode of the system. Moreover, invariant and guard conditions take the form of n-dimensional rectangles of $\mathbb{R}^n$. However, this definition barely justifies the presence of many clock variables, as their variation are synchronous and can not be separated. The only clear separation between the clocks comes from the reset function that may turn one or more clocks to zero without impacting the others. A theory of timed automata has been proposed in [30].

Timed automata give more tools to represent the behavior of systems by providing a representation of temporal evolution, which facilitates the representation and approximation of various types of systems [31].

There exist variations of timed automata, such as the stop-watch machines: this sub-category authorizes activating or turning off the different clocks of the system to synchronize them. In these automata, the condition $F(m, x) = 1$ no longer holds. It is rather replaced with another condition $F(m, x) = c$ with $c \in \{0, 1\}$.

From purely timed automata emerged a theory aiming at incorporating time as a variable in more complex automata with more non-clock variables [24].

It is, therefore, possible to express X as a concatenation of two subsets X_v and C with C a set of clocks following the corresponding constraints and X_v other continuous variables that do not respect the clock constraints. This allows the study of more complex systems and the combination of time constraints with more general variables of CPS.

In a timed automaton, the time elapsed between two discrete transitions is computed using the different clocks, which can be reinitialized to zero after their associated transition. Each transition is given a duration represented as its label d, and the upcoming of this transition generates a bound in the value of the duration clocks of d. If each transition is labeled with its duration time d_i and if the time elapsed between each transition is noted τ_i, then the trace of a timed automaton can be written $\tau_0 d_1 \tau_1 d_1 ...$.

Timed automata can be simplified by merging successive transitions and by adding their respective time lengths.

Different tools were developed to analyze and deal with timed automata [32, 33], proposing different approaches to manage the time parameters and evolution. One of the challenges in temporal automata is the bounding of the sojourn time, as explained in [34].

2.2.8 . Semantics

In computer sciences theory, semantics is the mathematical analysis of the meaning of programming languages and instructions. In the case of hybrid automata, the semantics represent the universe of its possible behaviors. According to [5], it is expressed using labels on the different transitions expressing its nature or its duration.

Definition 15 (Labeled Transition System) A labeled transition system S is composed of the elements

- A set of modes $\mathbf{Q}$ and two subsets $\mathbf{Q}_0$ and $\mathbf{Q}_{end} \subseteq \mathbf{Q}$ corresponding to the initial and terminal states of S.
- Σ a finite set of labels
- $\rightarrow \subseteq \mathbf{Q} * \Sigma * \mathbf{Q}$ a transition relation written $q_0 \overset{a}{\rightarrow} q_1$ or (q_0, a, q_1) for $(q_0, a, q_1) \in \rightarrow$

From the definition of a system and the knowledge of its state space, it is possible to use its dynamics to compute or write an execution of the system. The definition of the execution presented here has been proposed by [5].

Definition 16 (Execution of a discrete automaton) Let A be a discrete automaton (i.e., a hybrid automaton where $|X| = 0$). An execution of A is a sequence $(q_0, \langle q_i, a_i \rangle_{i \in [\![1,k]\!]})$ with $k \in \mathbb{N}$ and $\forall\, i \in [\![1, k]\!],\ q_i \in \mathbf{Q}$ and a_i labels characterizing the transitions.

If A is a temporized automaton, the labels a_i highlights the time elapsed during the transition $q_{i-1} \rightarrow q_i$.

The execution of a discrete automaton is a sequence of discrete states of the automaton state space separated by events (discrete transitions), which are labeled using elements from a chosen alphabet Σ.

This definition is adapted to different types of automata and can expose the discrete behavior of a hybrid automaton. The sequence of discrete states would then be replaced by a sequence of operating modes exposing the execution trajectory in the discrete space $\mathbf{Q}$. An execution corresponds to a specific trace of the system and highlights one of the potential trajectories authorized by the system dynamics.

In the case of hybrid systems, when $|X| \neq 0$, the knowledge of the continuous evolution of the variables of $|X|$ is lacking in the representation of the execution of the automaton.

Therefore, the management of X must be split into different modes, and studying global dynamics becomes more complicated.

The execution of a hybrid automaton has been presented in [35]. It requires the definition of an abstraction function $\alpha : \mathbf{Q} * \mathbb{K}^n \rightarrow S$ with S a countable set such that each value (m, x) of $Q * X$ can be associated with an element of S.

Definition 17 (Partition of a set) Let E be a set of elements. A partition of E is a set $\{E_i\}_{i \in [\![1,n]\!]}$ such that :

- $\forall (E_j, E_k) \in \{E_i\}^2, j \neq k \implies E_j \cap E_k = \varnothing.$
- $\bigcup_{i=1}^{n} E_i = E$

Definition 18 (Discretization/Abstraction function) Let S be an uncountable set. An abstraction function (also called discretization function) $\alpha : S \rightarrow \{S_i\}_{i \in [\![1,k]\!]}$ is a function mapping the set S to a finite set of subsets $S_i \subset S$ such that $\{S_i\}_{i \in [\![1,k]\!]}$ forms a partition of S.

The abstraction function is a method to visualize an uncountable set as a finite set of elements.

This notion of discretization will be essential in our works and constitute the basis of qualitative abstraction.

Definition 19 (Discrete transition system) Let there be a hybrid automaton H and a discretization function α defined as above on $Q * \mathbb{K}^n$. The discrete transition system corresponding to H is the system defined by $\langle Q \bigcup X, Init, T_\alpha \rangle$ sharing the same state spaces and initial states than H and with $T_\alpha = \langle (m_i, s_j) \rightarrow (m_k, s_l) \rangle$ such that $\forall\, (m, s) \rightarrow (m', s') \in T_\alpha$:

- either $m \neq m'$ if $m \rightarrow m' \in T$, which implies a discrete transition of the system
- or $m = m'$, which implies an intra-modal transition of the system. In that case, $s \neq s'$, and there exists a time step $\tau > 0$ and $x_i, x_j \in \mathbb{K}^n$ such that $\alpha(m, x_i) = s$, $x_j = x_i + F[m](x_i) * \tau$ and $\alpha(m, x_j) = s'$.

With this definition, we can now represent a hybrid automaton with a discrete system with a finite or countable quantity of transitions. Therefore, just as we defined an execution of a discrete system, it is now possible to define an execution for hybrid systems.

Definition 20 (Execution of a hybrid automaton) Let there be a timed hybrid automaton H, α a discretization function, and H_α the corresponding discrete transition system. An execution (or run) of H is a succession of transitions of discrete states of H_α : $(m_0, s_0) \to (m_1, s_1) \to \dots \to (m_k, s_k)$ where $(m_i, s_i) \in \mathbf{Q} * S\, \forall\, i \in [\![0, k]\!]$. For $j \in [\![0, k-1]\!]$, either $m_j \neq m_{j+1}$, or $m_j = m_{j+1}$ and $s_j \neq s_{j+1}$

Remark 2 *As was the case for discrete automata, it is possible to complete the execution with the labels characterizing each transition by noting them* $(m_i, s_i) \xrightarrow{l_i} (m_{i+1}, s_{i+1})$*. These labels can either characterize the nature of the transition and the time elapsed or provide any other knowledge about the event. It is, however, considered that, for a transition* $(m, s) \to (m', s')$*, if* $m \neq m'$*, then the time elapsed will always be* 0*. We consider the discrete transition instantaneous by analogy to step functions where the result instantaneously jumps between two values.*

The sequence of the labels of an execution h of H is called the trace of h. Considering that the labels of the transitions are taken in a set of labels L, the trace of any execution h of H is a word from $L^\star$.

Definition 21 (Accepting execution) Let $((m_i, s_i)_{i \in [\![0,k]\!]})$ be a run of the hybrid automaton H and S_t a set of states of H called the terminal states. If $(m_k, s_k) \in S_t$, $((m_i, s_i)_{i \in [\![0,k]\!]})$ is said to be an accepting execution/run of H.

From the notion of execution of a hybrid automaton, it is possible to define its semantics.

Definition 22 (Hybrid automaton semantics) Let there be H a hybrid automaton. The semantics of H, noted $[[H]]$, is the set of all the executions h of H.

As in [26], we limit the context of the presented works to cases that do not include so-called zeno behaviors (i.e., infinite behaviors happening in a finite time interval). In cases like the bouncing ball that is theoretically zeno, we make the simplification that a converging behavior reaches its convergence value once a chosen neighborhood is reached.

2.3 . Numerical Simulation

Numerical simulation was conceived as a computational technique to approximate the numeric solutions of mathematical models by discretizing the time scale involved in continuous equations, which are then solved with specific algorithms. Numerical simulation allows researchers to model and predict the behavior of complex systems that are difficult or impossible to analyze using traditional analytical methods. It involves approximating the solutions of mathematical models through the discretization of continuous equations, which are then solved using numerical algorithms.

The equations modeling a system (be it *ODE* or partial differential equations) and corresponding initially to continuous behaviors are discretized on the scale of time in order to represent the evolution of the variables between successive time steps.

From the simplest resolutions, such as Euler or Runge-Kutta resolution methods, to techniques, such as finite element or finite volume methods, they have in common the discretization process of the equations to process them computationally [36]. These methods require a careful balance between computational efficiency and accuracy, with considerations for numerical stability and error propagation [37].

Given a system whose dynamics is expressed by a flow condition F a continuous (Lipschitz) function and an initial condition X_0 such that $X(0) = X_0 \in \mathbf{X}$, numerical simulation consists in computing a trajectory $\Phi : \mathbb{R}^+ \to \mathbf{X}$ representing the anticipated evolution of the system variables from the initial state X_0. The result is a set of points abstracting the exact trajectory at specific time instants. These instants are set depending on the chosen integration step Δt, whose value will strongly influence the precision of the final result.

As numerical simulation is largely used for many fields and objectives, tools have been developed to adapt it to hybrid systems, such as in [38].

To observe the complete behavior tree of a system using numerical simulation, one must execute an important number of simulations, as a behavior may include an infinite number of disjointed trajectories. Moreover, if numerical simulation can prove the existence of some behavior, it cannot be used to demonstrate the safety of a system, as any quantity of numerical simulation could fail to exhibit a possible trajectory. However, numerical simulation is nonetheless vastly used in many industries for verification and certification purposes. There is, therefore, a major need for methods with a better formal basis, allowing formal tools to give more reliable certifications.

Moreover, numerical simulation poorly handles uncertainties. As real values are naturally replaced by float numbers to execute the simulation, uncertainties and approximation are an inherent part of the process, and managing them can create problems. To this extent, more general methods have been developed to handle this difficulty.

2.3.1 . Interval propagation

To handle the presence of initial uncertainty and to take into account the intrinsic uncertainty linked to the use of float computation, some contributions developed the use of constraint propagation. It consists of contracting interval domains associated with variables of $\mathbb{R}$ and applying the dynamics equations of a system not on numerical values but on numerical intervals or rectangles. It can be used to propagate uncertainties in the situation where errors are represented by intervals. Interval propagation considers the problem of estimating the value of a variable as a constraint satisfiability problem.

Apart from measurement errors, interval propagation allows the inclusion of many uncertainties from different sources and the representation of the level of knowledge available at any time instant about the value of a variable of the system.

2.3.2 . Flow-pipe methods

Like interval propagation techniques, flow-pipe methods were designed to propagate uncertainties with a more adapted representation than intervals/rectangles. As developed in [39], a significant drawback of interval propagation is the increasing size of the uncertainty rectangle while a simulation is running. More precisely, as explained in [40], the uncertainty rectangle will expand on sub-spaces of $\mathbf{X}$ on which the dynamics flow F is expansive (see def 23), meaning that a model can arise where the knowledge about the value of X has no interest at the end of an interval propagation.

Definition 23 (Contractive/expansive functions) Let there be a function $f : \mathbf{E_1} \to \mathbf{E_2}$ with $(\mathbf{E_1}, d_1)$ and $(\mathbf{E_1}, d_2)$ two metric spaces. f is said to be a contraction if f is k-Lipschitz with $k < 1$. Otherwise, f is expansive.

Flow-pipe has been designed as an over-approximation set of reachable states of a continuous dynamical or hybrid system given an initial set of states [39, 41, 42] to avoid the risk of having useless approximations at the end of the simulation in the case of extensive functions. It differs from the classic interval representation in the form of the uncertainty space. Rather than using a rectangle shape, flow-pipe, as presented in [39], uses a cylinder (a pipe) centered on the supposed value of the system and including an uncertainty neighborhood.

More specifically, in a situation where the knowledge on a variable x is $x \in [a, b]$ with $a, b \in \mathbb{R}$ and $a < b$, and if we have to solve the equation $x - x$, interval propagation would compute $[a, b] - [a, b] = [a - b, b - a]$ instead of returning 0. Therefore, as $x - x$ should leave no place for any uncertainty, interval propagation would create more uncertainty on $x - x$ than on x, which is a problem.

As interval propagation is a too naive approach to solve some simple equations, flow pipe uses affine arithmetic [43], consisting of expressing a bounded variable x not with a set of inequality nor with an interval, but using the affine form $x = x_0 + \sum_{i=1}^{m} \epsilon_i x_i$ with $\forall\, i \in [\![1, m]\!], \epsilon_i \in [-1, 1]$ noise symbols associated with the standard deviation x_i.

The use of affine arithmetic allows for safe approximations of functions that over-approximate the set of reachable states of a system given initial conditions and uncertainties.

From this initial knowledge, the flow pipe algorithm constructs a lower and an upper trajectory to confine the value in the obtained uncertainty borders, which grows and reduces depending on the local contractive or expansive property of the function.

As the computed derivatives of the trajectories are bounded into a certain flow-pipe, the trajectory itself can be bounded and evaluated with a certain degree of confidence.

In [40], the authors developed a simulation of CPS using a similar method. First, it splits the state space into areas where the flow functions are contractive or expansive. Then, the conceived tool propagates the system's state using topological balls.

The similarities between the approaches are noticeable as they all seek to propagate an exhaustive trajectory of the system from an initial state, taking into account the maximum deviation allowed by anticipated uncertainties.

As flow-pipe computation delivers two extremely pessimistic trajectories that correspond to the most abnormal behaviors, it can be used for property and safety verification. If the flow pipe does not intersect with the dangerous subsets of the state space, the system can be considered safe in standard use conditions.

Moreover, a topological study from [40] creates an abstraction of the state space based on a qualitatively expressible property (are the flow functions k-Lipschitz with $k < 1$?), showing the thematic and reasoning proximity with qualitative modeling. However, even a high degree of confidence cannot replace formal proof.

2.4 . Computer Algebra/Symbolic Computing

If numerical simulation is a well-developed and almost universal choice to provide approximate solutions to the numerical, it does not allow formal proof or symbolic resolution. Symbolic computing (or computer algebra) is a field of computer sciences aiming at manipulating formal mathematical expressions. Compared to classic computation and numerical reasoning, it allows more formal proofs by computing equations and formulas without numerical considerations and respecting symbolic mathematics rules.

If some contributions separate computer algebra from symbolic computing [44], the two terms are often associated to describe the representation of mathematical objects using symbolic representation, such as in [45].

Computer algebra does not allow any approximate representation of a value or a variable, as each mathematical object is represented in an exact and finite form. Therefore, rational numbers are represented in their fractional form, with real values represented symbolically.

Symbolic computing comes from analytic mathematics, and the development of the discipline in the last few decades allowed us to solve several problems, such as polynomial system resolution or cylindrical algebraic decomposition [46]. Symbolic computing could also be helpful in the improvement of SMT solving [47] as it offers powerful tools to solve complex constraints. However, it appears that the two procedures are, for now, still mainly separated. This can be illustrated in our works with two tools we happened to use, which are *Sympy* [48] and *Z3* [49]. The first focuses on symbolic computing, while the second focuses on SMT solving.

Symbolic computing can also be related to symbolic execution. Symbolic execution is defined in [50, 51] as a program analysis method to test and prove specific properties of a program by traveling through its execution tree using symbolic logic to cover every possible outcome of the program. If symbolic computing and execution are not to be confused, they share a core of conceptual analysis and are often bridged just as in [46]. Both can be used to prove properties using symbolic analysis of operations instead of relying on chance and examples.

2.5 . Satifiability Solvers

A solver is a tool that takes as input a mathematical formula modeling a problem and returns either a boolean characterizing if the given problem admits a solution or not or a solution to the problem if it exists. The solver generally takes as input a set of constraints written as mathematical predicates that limit the solutions to a subset of the variables' definition space. Given these constraints, the solver maximizes or minimizes a utility function representing the considered problem.

Some solving problems can be considered as satisfiability problems, while others focus on optimization. SAT problems are a category of problem that is interested only in the existence of a solution satisfying the different constraints passed as inputs in the solver but which do not care about optimizing this solution. A SAT problem is a decision problem aiming at determining if it exists a variable valuation such that a logic proposition is satisfied.

SAT solvers are boolean problem solvers who must deal with logic predicates and determine whether they can admit a solution or not.

For x, y two propositional values, a SAT solver with the logic predicate $\phi = (x \wedge y) \vee \neg y$ as input will answer $True$. On the opposite, the same solver with as input the predicate $(x \veebar y) \wedge (x \wedge y)$ will answer $False$.

2.5.1 . Satisfiability modulo theory

Satisfiability modulo Theories (SMT) solvers generalize the SAT solvers to more complex structures (or theories), authorizing the involvement of numerical variables, real numbers, or data structures. SMT solving couples a SAT solver with a solver of a given theory T to check the satisfiability of the formula involving such elements and interpreted in T. There exist various categories of SMT solvers with different associated reasoning theories. Some can reason with variables on $\mathbb{R}$ or $\mathbb{Z}$, while others are more suited to solve differential or polynomial equations.

Given a predicate ϕ with specified variables x_i, the goal of SMT solvers is to determine if there exists a valuation for all variables for which ϕ is satisfied.

For example, to the problem : find $x \in \mathbb{R}$ such that $\phi = \{x^3 + y > 0\}$ associated with the constraints $C = \{x > 0, y > 0\}$, a SAT solver will return $False$.

In our works, we mainly used the SMT solver *Z3*, but many others exist, such as CVC [52], MathSAT [53] or OpenSMT [54].

2.5.2 . Dynamic system proof tools

As reasoning on a dynamic system requires managing its evolution and dynamics, First-order logic is not expressive enough to allow an efficient representation of the system and to make formal verification of its properties.

However, higher-order logic paradigms such as Dynamic temporal logic can be used for such objectives. In [55], for example, the authors use dynamic temporal logic (which they define as a logic paradigm for deductive reasoning about hybrid systems, where the continuous dynamics are specified using a system of *ODE*s) to reason on differential invariants (the constraints on the results of *ODE* that will always be satisfied) to prove properties about a system and its dynamics.

This method is based on the concept of differential ghosts and Darboux polynomials to prove the invariants of the *ODE*. They made this choice to solve the problem of the difficulty of writing explicit solutions to equations that do not necessarily have one.

Some tools such as KeyMaera [56, 57] use this differential dynamic logic to model hybrid systems as hybrid programs and to prove complex properties about their behavior.

Hybrid programs, formally defined in [58, 59], use a syntax allowing operations including

- assignments (either resets or updates) noted $x := f(x, y)$
- control tests expressed as $?x > 0$

- differential equations relating the time derivative of a value to other variables $x' = -b$
- Control structures expressed using combinations of the previous elements related with the order operators $\cup, ;, *$

Using computer algebra, Keymaera intends to find explicit polynomial solutions to the dynamics differential equations expressed by hybrid programs whose properties are verified using symbolic decomposition. It can prove the correctness, safety, controllability, reactivity, and liveness properties of hybrid systems.

However, it appears that these proof capabilities mainly focus on proofs about the invariants of the dynamics of the system.

It is perfectly sufficient to demonstrate properties of safety and stability, but it may not be sufficient to visualize the different behaviors of the system.

On the contrary, qualitative reasoning should highlight every possible qualitative behavior and, therefore, prove more complex behavioral properties as well as safety, correctness, and stability.

2.6 . Common Sense Reasoning

Common sense reasoning [60] is a general reasoning paradigm aiming at reproducing the human ability to understand, deduce, and induce structural concepts without requiring formal or complex computation. It is inspired by the reasoning on common sense knowledge presented in psychology as the ability to execute reasoning operations without even considering it and with very limited use of our brain capacity [61, 62]. Common sense reasoning expresses the simple deduction and induction processes that occur daily in the human brain with minimal information and effort. On numerous aspects, common sense reasoning can be related to naive physics [63], which is the study of natural phenomena and laws of physics without consideration for equations or quantitative information.

2.6.1 . Origins of common sense reasoning

Today, "common sense reasoning" seems to have become a generic term for two distinct disciplines. Like many other computer terms, it has been borrowed from other fields, such as philosophy and psychology. In philosophy, allusions can be traced back to the Platonic vision of the world, illustrated by the allegory of the cave, showing the limits of simple human perception and simplified thinking based on excessive abstraction and partial knowledge of the world. Different definitions associated with various fields, such as psychology, metaphysics, and theology, developed throughout antiquity and the Middle Ages. In the 17th century, Descartes broke with these predecessors and equated common sense with the concept of *Bon sens* (literally good sense), i.e., the human capacity to understand and reason about everyday elements from a base of knowledge acquired either through experience, but above all through deduction from other logical predicates. This definition forms the basis of the rationalist thesis that places the human mind and its reason at the heart of reflection and understanding.

However, this idea includes the existence of fundamental logical axioms that cannot be demonstrated and must be humanly innate to serve as a basis for all further reason. It is these axioms, considered as the first notion of things, that Voltaire will, for his part, consider as common sense. This same idea, however, was to inform the philosophical trend opposed to that of the rationalists, empiricism, which placed experience above reason in human cognition. Humes and Locke, among others, supported these criticisms.

The presence of common sense in psychology goes back to Confucius and Aristotle, with reflections on the idea of metacognition enabling recursive, unconscious control of conscious cognitive processes. Much remains unclear about this recursive reasoning capacity and its implications.

Today, common sense is also a subject of study in neuroscience, as it is considered the foundation of human intelligence, and understanding its mechanism would enable major advances in research into consciousness and the biological nature of the human mind.

In computer sciences, common sense reasoning goes back to the earliest roots of Artificial Intelligence and the extensive work on symbolic AI [64] that led to the development of many expert systems. The main criticism of this early work was that these expert systems lacked generality and could not perform the supposedly simple tasks commonly attributed to human common sense. This difficulty is still attributed today to the difficulty of formulating elements of knowledge that seem evident to us in the form of logical and explicit axioms. This problem and inability to incorporate this common-sense reasoning were at the root of the first winter of AI, but also of the ideas behind the implementation of deep learning techniques based on neural network structures. Being unable to explain or express, it was vital to be able to reproduce. However, while these methods have increased the performance of systems capable of integrating large numbers of rules that cannot be explained by logical predicates, they still lack the capacity for abstraction, generalization, and adaptation that is inherent to common sense.

For the purposes of our study, the term "common sense" can be taken to mean human skills that one would want to implement in a computer system to make up for the shortcomings of today's machines.

2.6.2 . Common sense in computer sciences: motivations, explorations, and successes

In computer science, common-sense reasoning is primarily intended to address the current weaknesses of artificial intelligence programs [64]. In fact, although these programs are highly advanced, they are not always able to answer problems that seem simple or intuitive to humans. To paraphrase [64], in AI, "what is easy is hard". This means that the easier tasks seem to us, the more difficult they are likely to be to integrate into a program.

The difficulty today's AI programs have in grasping elements such as context, irony, emotions, or inconsistency makes it necessary to find alternatives capable of reasoning more humanely, applying simpler but more general rules, and compensating for these shortcomings.

Today, common-sense reasoning in computing has taken several different paths, which we will present in the next section, before attempting to give some applications and make the link with human logic.

2.6.3 . Logic reasoning and symbolic artificial intelligence

The basis of common sense in computer science goes back to before the first winter of AI when it was used to work on symbolic artificial intelligence. The basis of this idea is to represent the knowledge available at a time t in the form of logical predicates. These predicates are supposed to represent, as far as possible, what is known about a situation, a problem, or a context. This method has enabled the creation of expert systems specialized in resolving particular problems and has the advantage of being easily explicable, comprehensible, and deterministic. Using different forms of logic, such as combinatorial and modal logic, has made it possible to diversify and increase the complexity of the problems addressed. In addition, introducing fuzzy and multivariate logic allowed the resolution of more ambiguous situations that cannot be reduced to binary choices and knowledge. However, this technique soon came up against two significant difficulties. The first is a major lack of flexibility, observed in the total inability of expert systems to move beyond their very restricted resolution domain. This poses a problem, as a system of this kind can technically only be used for one application and is neither reproducible nor generalizable, thus rapidly losing interest in this type of solution.

The second was the massive amount of knowledge required to solve a single problem when it became complex: the need to enter all the logical predicates by hand made it unthinkable to use this type of system on a large scale. Combined with its lack of reproducibility and flexibility, this method of solving seemed to have little appeal. As a result, it is considered to be primarily responsible for AI's first winter. The main examples of work based on this method are the Jeopardy tool and the Cyc database [60, 64]. The former produced surprising results in its ambition to answer general knowledge questions but soon showed its limitations when answering more complex questions or solving uncertain problems. When it was first launched, Cyc was expected to be a general knowledge base capable of containing all human knowledge. However, the excessive number of predicates to be integrated into the system, the need to enter them manually, the difficulty of correcting false predicates, and the impossibility of achieving complete exhaustiveness made the project impossible. Even today, after millions of predicates have been added, the work is still far from complete. These difficulties inherent in the project have contributed to the advent of an AI winter and to convincing a majority of researchers of the merits of techniques such as the perceptron and neural networks in general. Although these past successes are no longer relevant, they are still worth mentioning as the symbolic representation of knowledge is still at the heart of many current and developing common-sense reasoning methods.

Today, progress has been made in various branches of the discipline, such as taxonomic, qualitative, and probabilistic reasoning.

2.6.4 . Taxonomic and ontological reasoning

The first successes of qu reasoning were achieved on the evolution of taxonomic reasoning and its extensions. A taxonomy is defined by a set of taxa (classes) and individuals and by their relationships. Taxonomic reasoning uses classification and relationships. The most common example of taxonomy is that of living organisms. This is divided into five taxa, themselves divided into phyla, and so on, down to the species level. Each taxon implies intrinsic properties that all associated individuals share unless otherwise stated. Taxons are thus organized in a tree structure, like inheritance in object-oriented programming. An individual is an instance of a class and inherits its internal attributes. A single individual can also be an instance of several classes at once, in some taxonomies more complex than the animal kingdom. Taxonomic reasoning, therefore, enables one to reason about the nature of the entities to manipulate and the concepts involved. This corresponds to the human tendency to think in terms of categories and to reason as much about the categories as the concepts themselves. As mentioned earlier, taxonomies have been extended, notably through ontologies, which are a more complete representation of the structures and contexts of systems, imposing constraints on relationships and categories.

Ontological reasoning, which is more comprehensive and elaborate, enables in-depth analysis and deduction of structures, the intrinsic properties of the objects processed, and the relationships between objects and their environment. Creating an ontology allows one to store a significant amount of knowledge about a situation and establish a framework within which structural reasoning can be validated. In other words, it provides a meta-model that can be used to reason about more concrete or applied models. An ontology is a model template containing a certain number of constraints that all models inheriting from it must verify and which allows partial and rapid deduction of general properties. In this way, one can not only define explicit constraints on a model but also manage its implicit constraints by deducing its taxonomic/ontological structure. By using this knowledge representation to good effect, an algorithm will then be able to answer basic questions about the nature of objects, such as "Between an ice cube and a lighter, which will be the coldest?". Work on ontologies is currently underway at CEA LIST, including contributions that integrate elements from other common sense reasoning paradigms.

2.6.5 . Common sense knowledge

Another way of reasoning about predefined knowledge is the one Google chose to create its Google Translate algorithm. This search algorithm scans a gigantic multidimensional graph of words to be translated and matches them with existing links in the graph: the closer words are in the graph, the more likely they are to be seen together. The result represents all available knowledge in the form of a graph that can be browsed to read it. The exponential increase in the size of such a graph with the complexity of the knowledge will make conventional graph exploration impossible for high-dimensional problems. Such a representation, if generalized to fields other than word processing, could be capable of representing an impressive number of concepts and the links between them, whether conceptual, semantic, or functional. The strength of this methodology can also be seen by comparing the results between the translation capabilities of Google Translate and those of algorithms using WordNet. WordNet is a similar network but built according to an ontological logic. This tool also enables algorithms to reason by analogy: given two similar contexts and behaviors, it will be possible to simplify the resolution of the second by starting from the results obtained on the first.

The main difficulty with this technique is the difficulty of setting it up. Indeed, to create a functional version of Google Translate, the company first had to set up a deep learning program and run it on millions of sentences found on the search engine in all the languages listed before creating this network. The resources required to develop a network of this kind are enormous. What's more, if one wants to create a knowledge network that is general enough to store all the knowledge a human could need, the resources required (both financial and temporal) would be prohibitive. The final difficulty is that the algorithm leading to the creation of this invaluable knowledge base is currently not explainable. This raises questions about its use and reliability.

However, by focusing the creation of such a network on a more restricted environment to answer a specific problem, it would be possible to target the learning of the deep neural network on a precise context or element. The solution would then lose in generality, but as the knowledge would be learned automatically, it would at least retain the modularity and reusability of the method. The problem remains, however, how to obtain the resources needed to carry out the initial learning process: as this involves web-mining to obtain the necessary quantity of data, one needs to be able to access all this data quickly enough and select the relevant ones, which in itself represents a non-negligible challenge. To create this concept of sub-networks specialized for a part of the problem and to avoid unnecessarily complex representations, one needs to determine what Hayes called the Conceptual Closure [65]. This involves determining the smallest possible general framework that can contain all the concepts required to model and solve a problem, which is difficult to achieve because exceptions can always arise. This requires much preliminary work to delimit the model design space.

2.6.6 . Temporal and causal reasoning

If one reflects on the limitations of the previous paragraphs, one of them is that they are primarily suited to static models and systems. They offer the possibility of analyzing a situation at a given point in time and deducing the relevant information, constraints, and properties. Still, they are insufficient for the analysis of a dynamic situation. Such systems can change their environment very quickly (which implies a change in exogenous constraints and relationships), as well as their operating mode (for example, in the case of a chemical relationship in which one of the reactants is consumed, the reaction cannot continue as before, and dynamics of the quantities of the different compounds will not remain the same). A theory of processes and dynamics was needed to reason dynamically. In Forbus's works [66], ideas and concepts for creating such a theory are presented using already mentioned elements (notably the creation of a network of acquired knowledge). However, Forbus did mention another aspect of processes, that of dynamic dependencies between the components of a system: given that the aim is to represent evolution, one needs to be able to know how a change in one of a system's components will impact the others. For example, to model a car correctly, it is essential to understand how pressing the accelerator pedal will modify the state of the wheels according to the speed actuated.

This aspect of common-sense reasoning is intertwined with simulation, which must be carried out on models that allow for dynamic evolution. Among the methods for representing functional dependencies, causal reasoning is significant. It is a way of graphically representing the implications of variation/movement between the components of a system, highlighting the causality between them. Causality [67] is an important element in human thinking since it enables us to anticipate the consequences of what we do and see.

Managing causality raises the question of managing time: while it can be simple to manage time in a numerical simulation, it is more complex to know what to do when moving away from it to focus on common sense. How do we integrate the time variable and the approximate computation of variations in causality reasoning? One of the options adopted so far has been to work only with key dates (also known as landmarks) but not with the intervals between them. This logic allows one to schedule events and reason about ranking/scheduling but is not effective when the ambition is to model evolution over time. This question, among others, was addressed by Slim Medimegh during his thesis at the CEA [68]. Although his aim was to get closer to qualitative reasoning, his work aligns more with the general common sense framework. In particular, he discussed modeling proportionality relationships in the broadest sense (including strict, causal, and inverse proportionality). It seems that by adding more precise information and extending proportionality to functional dependencies (quadratic proportionality, for example), we can outline the evolution between key dates, specifying the trajectory taken in the intervals. We can thus observe a trend in the intervals and values at key points. According to the vocabulary introduced by Forbus, a trajectory defines a history in which events (key points/landmarks) and epochs (intervals/rectangles) are involved. This idea underpins qualitative reasoning.

2.6.7 . Qualitative reasoning

Qualitative modeling aims to simplify real-world concepts and to abstract the values of variables, as well as their dependencies and dynamics.

The abstraction methods most commonly used in qualitative modeling are the causal representation of dependency relationships, the simple discretization of the space of variation according to specific properties (sign of variables, direction of variation, convexity, etc.), and the numerical analysis of remarkable properties of the system studied via the calculation of zeros of remarkable functions. These methods have already been studied by different contributions, Kuipers [69] and Tiwari [35] being the two references in the field. The second approach is closer to semi-qualitative methods since it requires numerical analysis. The subjectivity of common-sense reasoning here lies in the fact that models created from the same initial data but using two different abstraction methods will produce potentially different or even contradictory results. It should be noted that while this multiplicity of possible models allows for relatively permissive adaptation to the situation and objectives sought, it runs counter to the discipline's current drive for greater generality in computer models. It is, consequently, necessary not to choose abstraction functions without a method but to integrate contextual parameters into a higher-level model, automating the choice of abstraction method and, therefore, optimizing the choice of the ideal type of qualitative model.

This capacity for case-by-case adaptation would be an advance that could enable a single logical/algorithmic core to adapt to many situations.

This ability to reason in terms of quality offers a wide range of practical applications in design, proof, verification, diagnosis, and simulation. Studying a model by analyzing its qualities gives a higher-level, more general view of its behavior and enables us to verify different properties of its behavior. Moreover, the absence of numerical computation means that one can study the different behaviors of a system in greater depth and breadth, aiming for a certain exhaustiveness at the expense of precision. A prime application of this capability would be to explore a system's behavior from a set of starting points that could be grouped under one or more labels (qualities) and to reflect qualitatively on variations in the various variables to observe the full range of possible future trajectories. Combined with a cycle checker and a constraint solver, we can obtain an abstract computing method and reasoning capability that enables better observation and understanding of systems.

The main limitation of qualitative reasoning is the current difficulty of obtaining precise results. By its very nature, qualitative modeling sacrifices the precision of data and computation to earn generality. So, just as an accurate computation is not possible with a mental abstraction that is too vague, the observation of precise numerical properties is impossible with a qualitative model and will be all the more difficult to produce the higher the level of abstraction at which the model is built. The second intrinsic limitation of qualitative reasoning is the difficulty of performing operations on the qualities defining the model. Since they are no longer elements of $\mathbb{R}$, the elements of the space of qualities thus defined no longer necessarily respond to the various laws and operations usual to continuous spaces of variations. The best way to proceed is to ensure that the abstraction process defines an order relationship between the different qualities obtained. To do this, an idea would be to create a bijection between the partition of $\mathbb{R}^n$ thus obtained and a subset of $\mathbb{N}^n$: since qualitative partitions of state spaces are always countable, a bijection of this kind will always exist. Once this bijection has been created, we can establish relations and operations between subsets as equivalence on their image by the bijection. Although it does not represent actual behavior, this projection of the state-space partition onto a discretization of $\mathbb{R}^n$ allows to reproduce the simplification of calculations the human mind engages in to instinctively solve complex problems. Another limitation currently posed by qualitative reasoning as practiced is its adaptability not only to specific cases but also to the researched objectives. Depending on what we want to observe or prove, the most suitable model and the abstraction choices will not be the same. What's more, correctly choosing the number of qualities to use will result from optimizations based on situational and temporal constraints.

In addition, there are difficulties linked to the management of uncertainties: wanting to concretize qualities to obtain more information and knowledge, one would come up against the barrier represented by the one-way direction of knowledge variation. Information and precision can easily be lost but cannot be recovered without field measurement, external intervention, or probabilistic measurement of variation, which amounts to keeping numerical computations and removes the interest from the qualitative models. The solutions proposed to overcome these difficulties are to combine qualitative reasoning with other techniques, such as supervised semi-qualitative simulation to limit the loss of knowledge, the propagation of probability density functions to maintain a probabilistic distribution of variable values, or a supervised modeling element to modify the model in real-time to adapt it to changes in the environment as well as to the model's objectives. Information regain, however, remains an unanswered problem. Although knowledge loss can be limited at the cost of partial model quantification, losses cannot be prevented from adding up when operations follow one another. One way of attempting to regain knowledge is to randomly draw starting points in the current zone of uncertainty, simulate their behavior, and refine current knowledge. However, the only reliable way to compensate for the natural loss of knowledge is to carry out concrete measurements regularly or when uncertainty becomes harmful, which will naturally add precision to the state of the system at the measurement time.

In addition to purely qualitative reasoning, work has been carried out on so-called semi-qualitative modeling, combining numerical and qualitative elements in the same model. Examples include set propagation, which is at the forefront of interval propagation, and flow-pipe methods.

2.6.8 . Probabilistic reasoning

The various studies of system behavior using the preceding reasoning suggest a flaw: that of non-deterministic behavior (whether absolute or only at the level of abstraction we choose). A straightforward example is qualitative reasoning: suppose one drops a lead ball over a table and wishes to anticipate its behavior. Will it stop on the table or shatter under the kinetic energy accumulated during its fall? One way to answer this question is to represent the various possible behaviors in a probabilistic network, considering all possible options and weighting the different transitions with their probabilities. We can also apply this type of logic to causal reasoning. Rather than deterministic causality, one could consider stochastic causality, with only the influences of previous choices/events on future trends and events. This type of logic can be represented using Bayesian networks, considering weighted and uncertain causal possibilities.

In terms of application, the main apparent use of these methods is to model a system still at an early stage in its development or placed in a partially observed environment. The initial lack of information imposed by the situation means that we cannot rely on measurements or any model faithful to reality.

The weakness of these models will be their significant imprecision and the difficulty of relying on them alone to carry out a task: intrinsically, it will often be necessary to combine them with other models to give them a sufficient database or make up for the lack of knowledge they can provide.

2.6.9 . Analogy reasoning

Among Melanie Michell's works, one of her most important contributions deals with the implementation of an analogy-based reasoning module using subsymbolic AI systems [70, 71]. This was designed to solve problems of various kinds by comparing the current situation with a similar problem encountered in another context. The analogy is currently very limited in that the prerequisites for such reasoning are not the same in a problem dealing with language as in the resolution of a physical problem. The proposed solutions, therefore, still lack generality. The proposals currently accepted use a generalization of common-sense knowledge, matching concepts/situations by semantic, structural, or functional proximity. For example, to answer the question "What is for woman what king is for man?" it needs to perform a semantic proximity analysis to find the correct answer, whereas to perform an analogy from a physical problem to another problem already solved will require a structural and functional analysis.

It should be noted that neither the former nor the latter can stand alone: It will be easy to draw a false analogy between a solar system and an atom with a purely structural analysis, while a functional analysis will wrongly link a star to an oven. Reasoning by analogy aims to reproduce an innate human ability to make connections between very similar situations quickly. However, the human mind still has the advantage over algorithms in that it can quickly determine which criteria to base these connections on so as not to choose a wrong analogy. This is precisely the point on which we still need to make progress to improve the reasoning capabilities of the machines.

2.7 . Qualitative Abstraction

Definition 24 (Sign Algebra) Let us suppose that the comparison operators $<, >, =$ are defined on $\mathbb{K}$. The sign algebra on the field $\mathbb{K}$ is defined by the set of elements $S = \{-, 0, +\}$ and by the laws $(+, \times, .)$ such that $(S, +, \times)$ is a vector space with 0 the neutral element for the law $+$ and absorbent element for $\times$, and $+$ the neutral element for $\times$ and such that $\forall\, x \in \mathbb{K}, x.+ = x, x.- = -x$ and $x.0 = 0$.

Sign algebra is very popular in the abstraction process because of the various mathematical properties it verifies:

- Addition and product are commutative (i.e., $\forall\, a, b \in S, a + b = b + a$ and $a \times b = b \times a$)
- Addition and product are associative (i.e., $\forall\, a, b, c \in S, a + (b + c) = (a + b) + c$ and $a \times (b \times c) = (a \times b) \times c$)

The abstraction process corresponds to simplifying a model by isolating one or some of its characteristics and considering them independently from the rest of the model.

A classic example of qualitative abstraction is the variation table used to describe the behavior of functions evaluated on $\mathbb{R}$.

Example 4 *Let us consider the function*

$$f : \left\{ \begin{array}{c} [-\pi, \pi] \to [-1, 1] \\ x \mapsto sin(x) \end{array} \right.$$

Its behavior on its definition space can be either represented using a numerical computation (see Figure 2.2) or a table highlighting the sign of the function or its derivative on different sub-intervals of its variation space rather than showing the exact behavior (see Figure 2.3, Figure 2.4, Figure 2.5).

With this representation, we lose the information on the particular numerical values of the function for each point of its definition space, but we can represent more simply the information regarding its extrema, its sign, and the sign of its derivative.

Qualitatively abstracting functions allows the highlighting of specific characteristics of the application by removing unnecessary information from its representation. This better reveals the most pertinent elements of the functions by losing knowledge on many other aspects of the model. In the given example, the chosen qualitative abstraction puts forward different aspects of the functions, but in every given abstraction, we lose the numerical values for any valuation of x, and we, therefore, cannot make any difference between two numerical valuations that present the same qualitative properties.

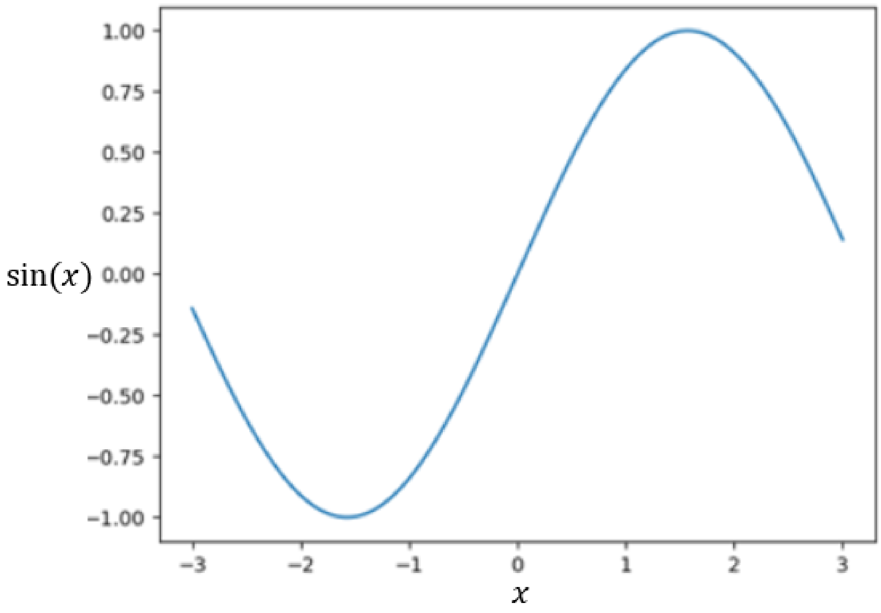

Figure 2.2: Numerical representation of the sine function

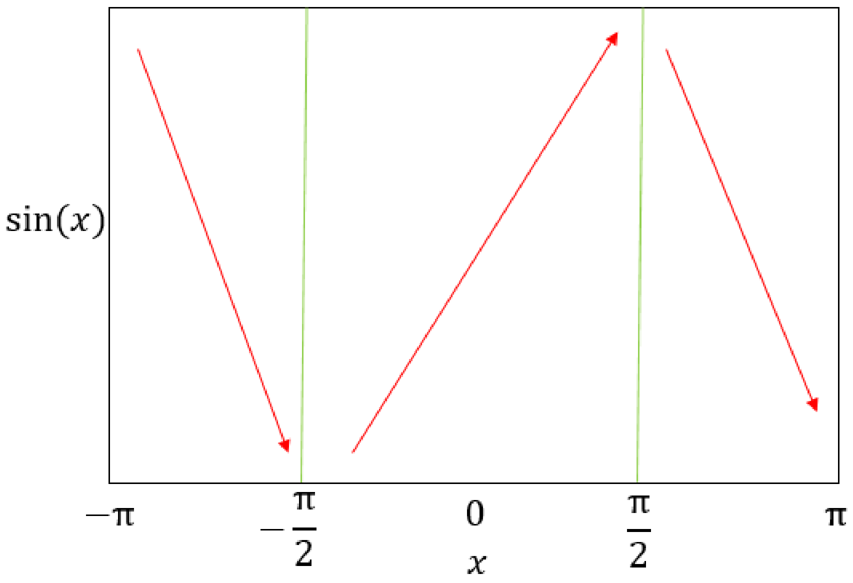

Figure 2.3: Qualitative representation (variations) of the sine function

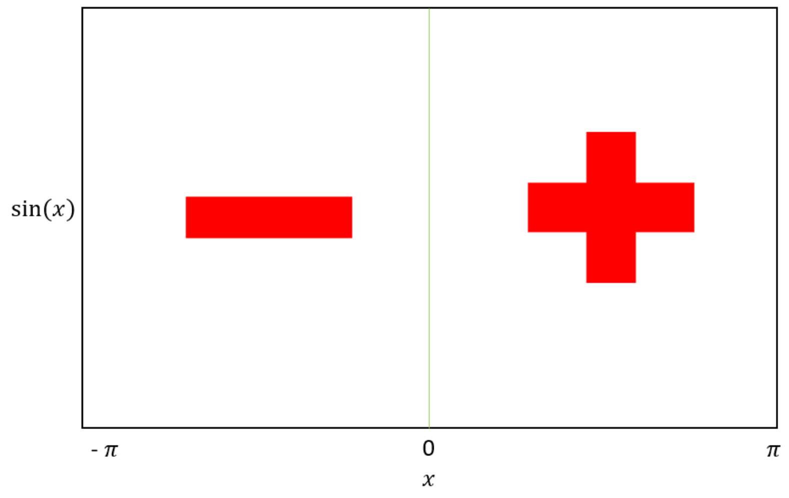

Figure 2.4: Qualitative representation (sign algebra) of the sine function

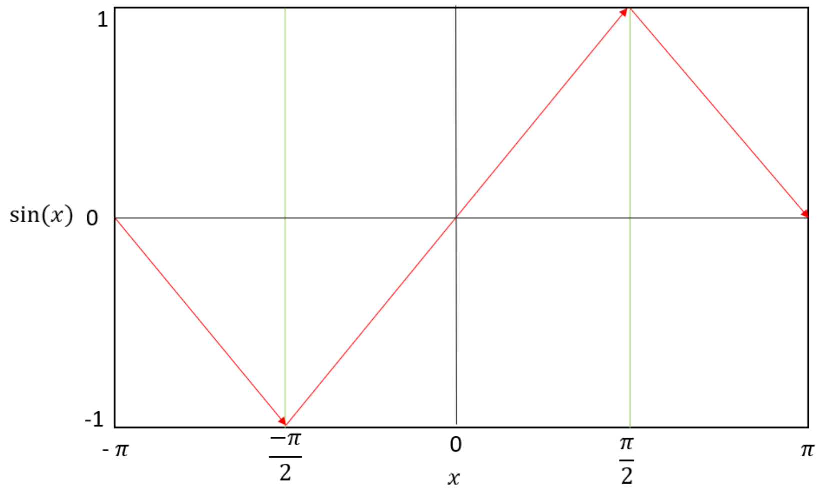

Figure 2.5: Qualitative representation of the sine function

Moreover, the diversity of abstracted representations shows that there is no unique method to qualitatively abstract the numerical models. Each numerical model can be abstracted in many ways in as many different representations depending on the desired goal.

3 - FRAMEWORK AND STATE OF THE ART

3.1 . Cyber-Physical Systems Modeling

3.1.1 . Stakes and challenges

CPSs are vastly used in many diverse domains, with more and more complex systems emerging to deal with the challenges of our time. If CPSs consist, by definition, of a software element controlling the physical processes of a system [72], it is more difficult to develop other common denominators to any instance of CPS. The number and the nature of the control mode may vary, the dynamics may be expressed using different forms, from *ODE*s to causality relations, and the guard conditions may be sufficient or necessary.

In the case of very complex systems such as nuclear power plants, every step of their lifetime is a major challenge and raises many technical, temporal, and financial challenges.

In the most critical cases, CPSs require to take up many challenges such as:

- accuracy [73] (to which extent does the modeled behavior correspond to the researched one?)
- Reachability (does the system design allow it to reach a specific state chosen as an objective?)
- Correctness (Which is a generalization of the previous challenge, with consideration for intermediate objectives and criteria)
- Reliability [74] (Can the system perform its task in every condition it may encounter?)
- Robustness (can the system perform with invalid inputs?)
- Understanding (implying the repeatability of a conception/action, the ability to use and modify the system)
- Safety (does the system risk to exhibit a dangerous or unwanted behavior?)
- Security (Is the system resilient to attacks/malicious behaviors?)
- Performance (How much time will the system take to achieve its goal?)
- Supervision (can we monitor the system efficiently to prevent, detect, and react to unwanted behaviors?)

3.1.2 . Modeling methods

To meet the stakes raised by CPS modeling in a more connected and industrialized world, various modeling and simulation methods, languages, and paradigms have been developed throughout the years to improve the representation relevance of specific aspects of the systems.

In [75], the authors inventory a list of various methods to model the CPS depending on the nature of the system and the goals of the model. Among them are UML modeling, formal-based methods, multi-modeling techniques, and component-oriented modeling.

Moreover, CPS can be specified using different syntaxes based on different natures of properties.

The most common method to represent CPS is to use a hybrid automaton, as we do to represent the sequence of modes and states allowed by the system's dynamics. They allow the representation of mode change using discrete jumps between the nodes of the automaton.

It is also possible to use other structures, such as Petri nets (representing the information and log flow in a system) or bound graphs (representing the transfers between the system elements and, therefore, the energy movements and contacts in the CPS).

In the industry, CPSs are often represented using tools such as Modelica and Simulink, both having their own application field.

Simulink is mainly used to launch numerical simulations, while Modelica offers more possibilities to reason on the structure and dynamics of the systems with an object-oriented representation.

3.1.3 . Current limitations

Most of these paradigms are specialized in some specific tasks and are less reliable for others. For example, Simulink is particularly fitted for simulation and system design and control but not very adapted for symbolic verification or formal proofs. It can be used to prove the system's robustness, but not their safety or to deeply understand their fundamental behavior.

Ontological reasoning allows structural detection of misconceptions in the system but cannot afford any verification on a more application level. It gives a high-level overview of the structures and allows us to prove their safety and reliability, but not their resilience. Testing and observing their behavior and the effect of borderline case inputs on the system is impossible.

In the case of very complex systems, and in the presence of systems of systems, having an exhaustive overview of the system's behavior, testing the different trajectories, and having a structure that allows an in-depth comprehension of its nature and operation is something that cannot be realized with only one modeling choice.

For example, the nuclear power plant case is striking. The ontological representation of this system gives a good overview of its nature, possible structural inconsistencies, and errors. However, it does not offer the possibility to test its behavior in real conditions. On the contrary, using numerical simulations to explore all the possible behaviors of the system takes much time and shuts the system down for an important period, which creates a visible loss of resources.

As an interesting compromise between the two levels of abstractions proposed by ontological reasoning and numerical simulation, qualitative reasoning offers a path to apprehend the study of CPS by the analysis of families of behaviors that share qualitative characteristics and allows to demonstrate the robustness and reliability of a system by symbolic computing rather than employing an important number of simulations to show a representative set of executions to show the robustness of the CPS empirically.

3.2 . Design Space Exploration

Design Space Exploration (DSE) refers to the analysis and selection of design points based on parameters of interest. In the domain of CPS, it corresponds more especially to the exploration and analysis of the parameters definition space (also called design space) in order to determine which of the possible values can be authorized.

DSE occurs at very early stages of CPS designs, far before numerical simulation, diagnosis, or monitoring, and is a critical domain of the conception of complex systems. The stake of DSE is to choose authorized values for the system parameters to satisfy some constraints on its behavior. It consists in selecting, testing, and comparing different configurations of values for the parameters of the system before its construction to verify some properties or optimize some utility function [76].

High-level DSE can be considered either as a multi-objective optimization problem [77] (as we may want to optimize a set of conflicting constraints and utility functions) or as a satisfiability problem if we only care about the boolean value of the constraint predicates.

The solutions of the parameters are often searched in a finite space of authorized values [78], and the arising problem consists both in the explosion of the number of possible solutions when taking into account all the parameters and in the critical time needed to test the validity of all of these solutions.

According to [79], an effective DSE requires three elements: a representation, an analysis, and an exploration method. The representation of the system must be considered to optimize the research, test, and comparison of solutions. However, working with CPSs represented by hybrid automata implies that the flexibility of the representation is limited.

The analysis requires a metric, a utility function, or a set of logic predicates to evaluate and compare the different valuations of the system's parameters. These elements correspond to the constraints and requirements imposed on the system that the parameters must satisfy.

Secondly, the exploration method is necessary to choose the valuations to be tested and will strongly depend on the nature of the design space.

Here, the difficulty will come from the fact that works on DSE improvement mainly work with finite design sets. However, at the first stages of a CPS design, there is no evidence that the prior knowledge of the system will be sufficient to reduce the design space $\mathbf{P}$ to a finite or even a countable set.

Finally, the choice of adapted values imposes a verification process to check the validity of the proposed solutions. The developed methods mainly consist of simulations to verify the constraint satisfaction with the chosen values. This situation poses two main problems: first, this is once more not adapted for infinite and even more for uncountable design space as it would be necessary to test the validity of an infinite number of values. Secondly, the presence of state variables in the system creates a situation where there are different sets of variables to deal with, which do not evolve and can not be treated at the same semantic level. Parameters are variables in the sense of an optimization problem and are constant in the referential of the state variables, while these state variables are variables in the sense of system modeling. The choice of the tools and the constraints will then be an important element in solving this challenging early-stage DSE.

3.3 . Stochastic Hybrid Systems

In [80, 81], the authors define stochastic hybrid systems (SHS) as a particular family of hybrid systems where the flow expressions are expressed as

$$\forall (m, X_i) \in \mathbf{Q} * X, \dot{X}_i = f(X, m) + g(X, m)$$

with f a deterministic function of X and m, and g a stochastic function of the same parameters. The discrete transitions are mapped and defined using stochastic conditions. Instead of following *ODE*, the system dynamics are written using stochastic differential equations.

In this contribution, discrete transition conditions and resets are united in transition maps. SHS are used to model systems and situations that cannot assure to give a deterministic response to the same situation. The authors take the example of a transmission control protocol where congestion windows or control systems cannot be represented using deterministic systems and, therefore, require the use of SHS.

In the literature, SHS are studied using truncated system representations by abstracting the expression of their dynamics to get more convenient equations for their analysis or by discretizing the system state space depending on the behavior of the dynamic equations and the nature of the probability density functions encapsulating the stochastic terms. This discretization can be achieved using Markov Chains or Markov Models [80].

Theories on the robustness and stability of SHS [82] have been developed to manage the impact of stochastic terms on the already complex structure of hybrid systems, with theorems such as Lyapunov's.

SHSs are particularly interesting for modeling hybrid systems evolving in partially known or completely uncertain environments. This mainly happens in communications systems, biological environments, or economics and financial modeling. All these systems have in common their various and important uncertainties that do not allow them to be represented as classical hybrid systems.

Example 5 *If we consider a heating system to which we add random disturbances, we can write it as a SHS with*

- $\mathbf{Q} = \{on, off\}$
- $\mathbf{X} = \mathbb{R}^+$
- $I = \{on : \{T_{env} < 100\},\ off : \{T_{int} > 60\}\}$
- $F = \{on : \dot{T}_{env} = h - c(T_{int} - T_{ext}) + \sigma w(t),\ off : -c(T_{int} - T_{ext}) + \sigma w(t)\}$ *with* h *the heating rate,* c *the cooling rate,* σ *the standard deviation of the random disturbances and* w *a white noise term representing these disturbances.*
- $T = \{(on, T_{int} > T_{up}, off, T \mapsto T), (off, T_{int} < T_{low}, on, T \mapsto T)\}$ *with* T_{low} *and* T_{up} *two thresholds.*

This system includes random terms that add the stochastic aspect to the model and exclude any use of classical CPS modeling on this system, as the dynamics and discrete transitions are not deterministic.

3.4 . Qualitative Reasoning

As presented in subsection 2.6.7, qualitative reasoning is a field of computer sciences (some may say artificial intelligence) aiming at representing, reasoning, and solving problems with fewer and lighter numerical information than classical resolution methods. Depending on the situation, qualitative reasoning can aim at completing, backing, or even replacing numerical reasoning in different algorithms. Qualitative reasoning includes:

- The definition and selection of qualities to be used in qualitative modeling.

- The abstraction of numerical values to these different qualities.
- The propagation of these qualities using qualitative simulation.
- The description and explanation of the obtained behavior using the available description of the system.

This section will present the motivations, the history, and the concepts of qualitative reasoning. This will allow us to introduce more precisely the stakes of our works as an improvement to the development of reasoning techniques.

3.4.1 . Stakes and motivations

Some physical problems involving many components, variables, and relations can generate various difficulties during a modeling, simulation, or solving process. The complexity of the system, as presented in [1], can be a barrier to its representation; the multiple configurations may cause a loss of generality for each execution, and the lack of information may prevent any general solution from being found.

Moreover, the computation time imposed by many physical problems is significant, which can be annoying in the presence of a time constraint.

The relations between the different components of the system may not be expressed using quantitative differential equations and, therefore, may imply a less precise link and formula not adapted to quantitative data.

Finally, a numerical model does not adapt well to subtle modifications in the system's core. Such changes would impose to completely rebuild the obtained model. Such vulnerability to change is a problem as systems often interact with their environment and cannot be completely protected from degradation or fatigue.

Qualitative modeling and reasoning appear as a solution to avoid or limit these problems by using more general, less costly, and more adaptable representations given by the qualitative paradigm.

The next section will introduce the emergence of qualitative reasoning in computer sciences and its development as a distinct research field.

3.4.2 . History of qualitative reasoning and modeling

Qualitative reasoning refers to the area of computer sciences aiming at representing and reasoning on models with very little and imprecise knowledge [66]. In a complementary way, qualitative modeling corresponds to the design of models adapted to support qualitative reasoning [83]. Originally, qualitative modeling was introduced by Brown [84] and De Kleer [85], who developed the concept of qualitative knowledge about systems and processes. They introduced this idea to represent knowledge that could or should not be expressed quantitatively. They designed this paradigm mainly for electronics and computer-assisted physics computation. They did not see 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 classical 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 require them. Since the beginning, qualitative representation has been seen as a tool for simplification to make the representation of physical models more convenient and general and to optimize the use of computational resources for the models and problems that would really require them. This need to reduce computational costs is even more understandable as computers did not allow heavy computations at the time.

If the notions implied by qualitative reasoning have evolved since then, the ambition remains unchanged: proposing a new reasoning tool to supplement the exact 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 [86, 63], the theory of dynamic processes [66], and the concept of conceptual closure of such a theory [65]. Qualitative modeling has made significant advances with the works of Kuipers [69] 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 [87]. 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 solved these problems with their work on semi-qualitative reasoning [88]. 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 [89], but the results did not meet the expectations. However, interval propagation has since really progressed, allowing advances such as flow-pipe computation [39] or more precise uncertainty measures and correction. Tiwari completed this approach [35] 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 considers the links between the dynamics and the values of a system, while previous methods had the drawback of separating the two aspects.

3.4.3 . Applications

As explained in the previous paragraph, qualitative reasoning has been developed for physical problem modeling and solving. If the initial application was to save computation time on complex problems, the current state of qualitative reasoning allows various different applications in different domains. Among these applications, we can present, for example:

- The prediction of the behavior of a system, using techniques to propagate qualities. Qualitative reasoning allows us to visualize all the possible trajectories and states of a system according to its known characteristics [66].
- The understanding of observed behavior, using the structure of the system and its observed trajectory [90].
- Formal computation and proof of structural or behavioral properties [91]: predicted sets of future states and wisely chosen discretization of the state space make it possible to prove properties of the system and improve confidence in its safety.
- The test of design constraints on qualitatively generated behaviors [92].
- Property verification [93, 6], which is complementary to the previous point.
- The deduction of the hidden information on a partially observed system.
- Approximation [31], allowing to observe a simplified and more convenient representation of a system or of its behavior.
- Fault diagnosis, using reverse propagation in the system to deduce the component or structure of the system that caused the fault [34, 94].
- System Design: On the contrary, qualitative reasoning applied at an earlier stage should allow us to determine the system configuration that would cause undesired behaviors or even faults. This, however, requires the expression of a set of constraints on the desired behavior of the system. This application is strongly related to DSE presented in section 3.2.

More generally, qualitative reasoning has or can find applications in many scientific and industrial fields: as the presented applications can be found in many disciplines, they bring with them the interest and the use potential of qualitative reasoning. Among these disciplines, we can find:

- Economy: As many parameters are completely or partially unknown in discipline, economic and financial computation are often either very complex or very imprecise (when not entirely false).
- Sociology: Qualitative reasoning could allow the modeling of very large-scale behaviors without having to consider individual variations and improve modeling and reasoning methods on human factors that are still hazardous and imprecise.
- CPS Design and test [95, 96].
- Chemistry and biology.
- Predictive Maintenance: With the ability to predict the normal conditions behavior of a system, it is possible to detect very small behavioral deviations of the execution and deduce likely problems to come and, therefore, the best corrections and active maintenance to do.

In our works, we mainly focused on application fields surrounding CPS, such as system design, simulation control, real-time system monitoring, and proof of properties. These different use cases will be detailed further in this document.

3.4.4 . Intuitive idea

As implied in its name, qualitative reasoning (resp. modeling, simulation) can be opposed to quantitative/numerical reasoning (resp. modeling, simulation). The opposition between qualitative and quantitative requires avoiding as much as possible numerical considerations if choosing qualitative versions. Literally speaking, qualitative reasoning refers to the use of qualities instead of quantities and, therefore, the loss of numerical and precise information about the system and its components.

Example 6 *To illustrate the difference between the two reasoning paradigms, let us consider the example of an object of mass m falling from a given height to the ground (reference height of 0). At each time t, the height of the ball is noted h and its speed $\dot{h}$.*

To reason on the different parameters and variables m, h and $\dot{h}$, one can consider their numerical value (for example, $m = 1.5$ kg, $h_0 = 3\,m$ and $\dot{h}_0 = 0\,m.s^{-1}$). On the contrary, dealing with qualities imposes the definition of categories in the state space of each parameter/variable in order to replace numerical values with *these categories (the so-called qualities). Instead of using precise values, one can consider its weight (resp. height, speed) in a pre-chosen scale, which gives less precise but still useful information about the value. For example, the qualitative value of the mass could be expressed among the finite set of labels* $Q_w = \{\textit{light}, \textit{medium}, \textit{heavy}\}$*, the position with labels from* $Q_h = \{\textit{very low}, \textit{low}, \textit{high}, \textit{very high}\}$ *and the speed with elements from* $Q_s = \{\textit{slow positive}, \textit{slow negative}, \textit{fast positive}, \textit{fast negative}\}$*.*

As presented in this example, the concept of qualitative reasoning implies suppressing numerical values defined on infinite or even uncountable sets with labels from a finite set. These labels must be chosen upstream in the modeling process depending on the desired balance between precision and simplification.

Transition from quantitative to qualitative: As qualitative reasoning implies fitting finite labels to possibly uncountable values, choosing the qualities is equivalent to abstracting the quantitative state space in a finite set. The obtained labels are abstractions of the numerical values. This means that there exists an abstraction function as defined in definition 18 that maps each element of the initial continuous state space $\mathbf{X}$ of the variable x to the finite set of labels Q_x such that every valuation of x is associated a label to reason on. On the example presented above, using qualitative reasoning requires to instantiate the sets Q_h, Q_s and Q_w and to define abstraction functions $\alpha_h : \mathbb{R}^+ \to Q_h$, $\alpha_w : \mathbb{R}^{+*} \to Q_w$ and $\alpha_s : \mathbb{R} \to Q_s$ mapping the continuous state spaces to the corresponding abstract labels.

The abstraction function should be injective: the set of qualitative labels should be designed to split the numerical state space by keeping as much knowledge as possible and by making it more intuitively accessible.

Therefore, the use of qualitative reasoning is often accessible in numerical systems as long as it is possible to define qualities that include and characterize intrinsic information about the variable and its values.

However, abstracting implies a loss of knowledge as the set of qualitative labels includes far fewer elements than the numerical one. Consequently, taking the reverse path is often impossible as there is no reliable method to discriminate two different numerical values associated with the same qualitative label once the discretization has been achieved. This means that some decisions may have to be taken without complete access to the information about the system. Hence, it is crucial to optimally define and select the discrete set of labels for each variable, as these qualities will have to be sufficiently explicit to reason without risking dangerous errors. The action of transforming an abstract quality to a precise value or a set of such numerical valuations is called concretization [35].

Definition 25 (Concretization Function) Let S be an uncountable set, Q_l a set of qualitative labels, and $\alpha : S \to Q_l$ an abstraction function mapping each element of S. A concretization function $\beta : Q_l \to P(S)$ is a function mapping each element of Q_l to a subset of S.

The next paragraphs will present the classical qualitative reasoning methods with simple case studies and will highlight the limitations of the current approaches.

3.4.5 . Methods and examples

This section will detail different methods used to operate qualitative reasoning in computer sciences. These various techniques have been developed to solve different conceptual or concrete scientific problems posed at different moments of the evolution of computers and computer sciences.

Causal Reasoning and Envisionment

Causal reasoning is a major part of qualitative reasoning and is based on the causality principle, ordering events between causes and consequences [87, 89]. Causality is an important concept to describe, explain, and understand physical phenomena. Causality can explain how a system works based on its structure by connecting every possible modification in the system to a set of associated consequences.

At the early stages of design and for non-specialists, many relations and structures are represented using causality dependencies.

Causal reasoning aims to explain and anticipate the behavior of physical systems using these dependencies.

In the literature, causal reasoning can sometimes be mixed with envisionment.

Proposed by De Kleer [85] and constituting the root of modern qualitative reasoning, envisionment is a qualitative reasoning paradigm aiming at computing all the evolution possibilities of a system from a given initial point in order to anticipate every possible behavior of the system. The obtained trajectories are then represented as a tree, which is supposed to include every possible trajectory of the system in its state space.

Envisionment requires discretizing the system's state space in subspaces and representing them as symbols to manage the evolution of the system variables among this set of subsets. The tool *Garp3* [97] bases its structure and exploration on the principle of envisionment by studying the dynamic equations of the system.

Qualitative reasoning is a more specific field of study that only focuses on the concepts of causes and consequences. In computer sciences, two main types of causal reasoning have been defined: mythical causality [98] and causal ordering [99, 67].

De Kleer and Brown defined mythical causality as a way to explain and prove the behavior of a system using qualitative state diagrams. Such a diagram exhibits both all the qualitative states of a system and the transitions between them. Such a diagram is unique. It incorporates the different causality properties, such as the unicity of causes, the temporal ordering between causes and consequences (a consequence comes after its cause), and the structural proximity between cause and consequence. This diagram exhibits causal relations that explain transitions between qualitative states. However, such a graph cannot deal with causality and evolution inside the different qualitative states. Among other problems, classic time representation does not allow such reasoning to be used inside of qualitative states.

To solve the weakness of the classic impossibility of having a causality reasoning inside of a qualitative state, the authors introduce the concept of mythical time, which introduces a partial ordering between events and which has no physical existence (i.e., a positive mythical duration will not last more than an instant in real physical time). Over mythical time, the different confluences and equations of the system may be violated in order to authorize the propagation of a perturbation variable by variable. A period of mythical time is then used at each transition of the system to compute the change of state and the rebalancing of the system.

Moreover, mythical causality allows us to determine and take into account the presence of a feedback loop in the system.

On the other hand, causal ordering, as practiced by Iwasaki and Simon, consists of a structural analysis of the system more than a symbolic computation of the equations in a graph. It links the variables and the equations of the system to exhibit the dependence between them.

The obtained structure shows the influences of every element of the system on the others. However, it does not consider the mathematical resolution of the mathematical equations to express the temporal ordering of the events of the system.

For example, an evaporator system presented in [99] shows the causal ordering among the variables of the system, which looks like Figure 3.1 where each arrow represents a causal influence from the source variable to its target.

Qualitative Process Theory

Forbus has introduced the Qualitative Process Theory (QPT) in [66]. He defined it as a language designed to write dynamical theories to represent systems' evolution. Qualitative process directly derives from the general concept

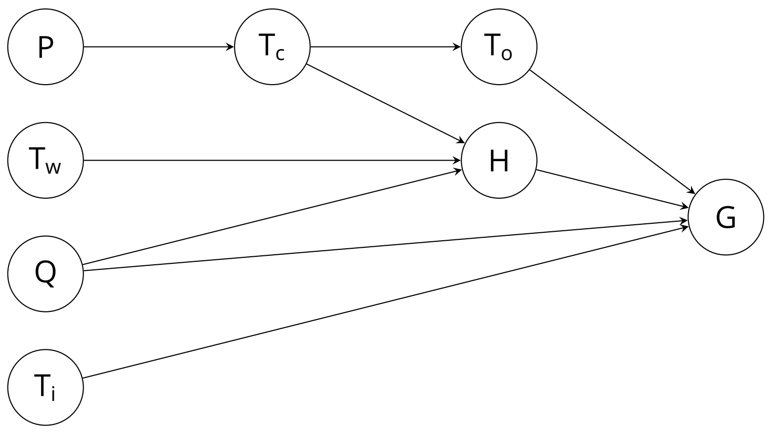

Figure 3.1: Example of a causal ordering representation

of process, defined as any element that introduces a change in the state of the studied system.

According to Forbus, reasoning on orders of magnitude, as explained in section 3.6, is insufficient to claim explainability and reliability in studying systems and processes.

Moreover, if the classic and intuitive representation of system perturbations as logic predicates is efficient in small structures, it is not a relevant solution in more complex systems as the number of resultant predicates may explode quickly.

As in current literature, he uses states to represent the system's situation at a current instant. However, he separates the continuous states corresponding to **episodes** from the punctual and instantaneous states corresponding to **events** and that separates the tendencies.

To represent the position and evolution of its system in order to introduce its QPT, Forbus defines the opposite concepts of **situation** and **history**. A situation is restricted in time but not in space: it represents the state of the system on a limited time period. On the contrary, a history is bounded in space but not in time: it represents the evolution of a system on a limited state space during any interval of time.

The introduction of a process theory aimed to answer the question of the effect of the interactions of the different components of a system and to understand the dynamics implied by its structure and by the different perturbations of the conditions.

The representation of influences between the components of a system must allow one to deduce complex processes between many objects and express the actual dynamics linking them. Moreover, a complete theory representing both direct and indirect influences gives knowledge about every possible cause of change for each component and should allow the addition of these influences to anticipate or explain the evolution of a component exposed to multiple influences. However, this multiplicity and the loss of information implied by qualitative reasoning create an important uncertainty about the relative weights of the different influences applied to a single component. The represented system is, therefore, not always deterministic, and this is one of the major weaknesses of this process theory.

The management of temporal aspects can be achieved either using envisionment as presented in subsubsection 3.4.5 or using numerical simulation, depending on the precision requirements.

The main limit highlighted in [66] is that to generalize this contribution to various physical systems, each category of systems should have its own grammar and syntax, as it is not possible to give the same template of influences, rules, and predicates for mechanical, chemical or electric systems. The QPT, as presented by Forbus, was seen as a contribution to developing further qualitative reasoning capabilities. As he explained after the development of a qualitative version of a classic physical demonstration, "It will be interesting to see what other results from the classical theory of differential equations can be derived from qualitative information alone."

Naive Physics

Initially, naive physics seems to date back to the time of Aristotle and its philosophical school of scholastic [100]. It can be seen as the study of the human perception and comprehension of simple physical phenomena. A modern version of naive physics [86, 65] has been introduced by Patrick J. Hayes in his naive physics manifesto and corresponds to an application to the field of computer sciences of the original naive physics discipline. In this context, naive physics aims at formalizing the common knowledge of human beings to improve the understanding and reasoning capabilities of machines.

For DiSessa [101], naive physics knowledge consists of "a fragmented collection of ideas, loosely connected and reinforcing, having none of the commitment or systematicity that one attributes to theories”. This vision strongly opposes the first one as it denies the existence of a complete theory of naive physics and, therefore, implies that no general language can unify every aspect of common sense reasoning.

Hayes aimed to formalize elements, from common sense reasoning to reasoning about situations and systems. In contrast with other scientists who sought to develop a general program using their formalism to upgrade the state of the art with better performances, he considered that developing such an algorithm would be at least pointless and, at worst, dangerous.

A major contribution of the naive physics manifesto is the concept of **conceptual closure**. It illustrates the smallest set of concepts required to express the relations, objects, variables, and situations that intervene in the system.

A model verifying a conceptual closure will only present dynamics and situations that the chosen concept will be able to represent. The certification of conceptual closure would be, in theory, very complicated to achieve as exceptional situations can always happen, but such a validation would be particularly convenient to ensure the reliability of the theoretic background. The problem comes from the inherent conflict between closure and breadth. A valid theory could then be represented as a connected graph linking the different concepts to each other.

His idea is to define each physical system as a system of systems and create a set of sub-systems containing their own relations, definitions, and concepts.

The objects would then be linked using causality, dependencies, and composition relations.

Since then, naive physics has been an object for more philosophical debates or epistemological studies, such as in [102]. However, the computational field seems to have been progressively abandoned.

Example 7 *Let us take, for example, the physical phenomenon of gravitational forces. Describing such physical force can be achieved at different precision levels. First, the most correct, reliable, and complex way to represent gravitational interaction is to use relativity theorems and, more significantly, general relativity as illustrated in Figure 3.2. However, the complexity of this theory implies that neither scientists nor ordinary people use this model in general. In most cases, the first will instead use the Newtonian model, such as represented in Figure 3.3, while untaught people will likely represent the phenomenon using more Aristotelian representation only involving the mass of the involved objects Figure 3.4.*

These different levels of representation illustrate that the needed level of precision/complexity trade is not the same depending on the researched goal. Naive physics and, more generally, qualitative reasoning aim at developing a reasoning paradigm and a base of work for cases where precision is superfluous and high degrees of complexity are to be avoided.

Qualitative Modeling and Abstraction

In many contributions of the literature [35, 13, 87], the main step for creating a qualitative model from a system or its numerical representation is to abstract the state space of the system. This requires the use of an abstraction function as presented earlier in subsection 3.4.4.

From early in the development of qualitative reasoning [69], qualitative abstraction has been improved using different concepts.

In [66] and before, the discretization of the state space was achieved using predicates describing properties of the system.

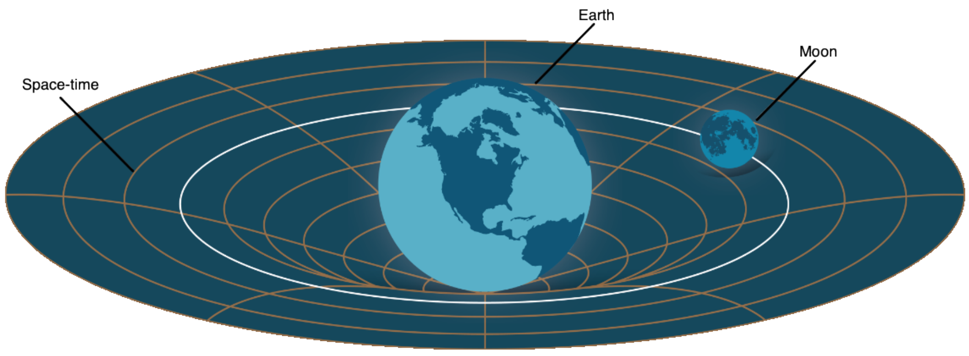

Figure 3.2: Real Gravity Forces (according to general relativity)

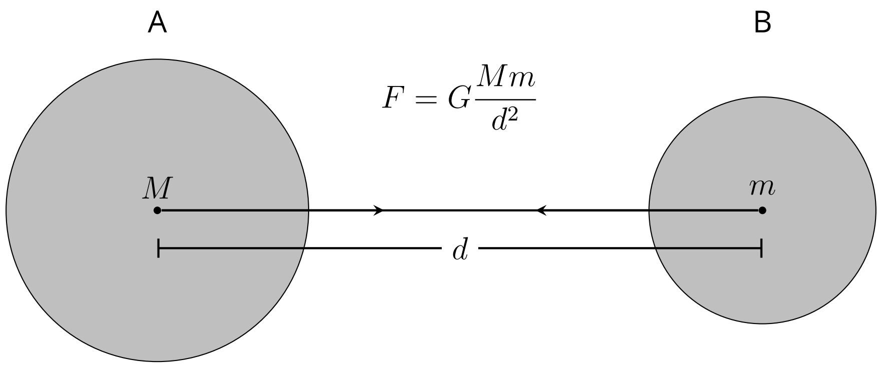

Figure 3.3: Newtonian Gravity Forces

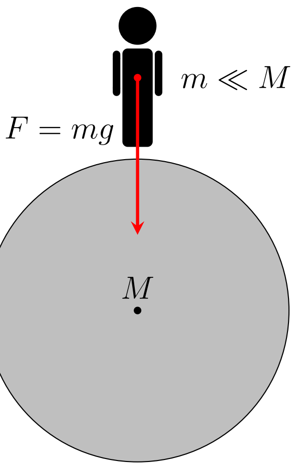

Figure 3.4: Simple Gravity Forces

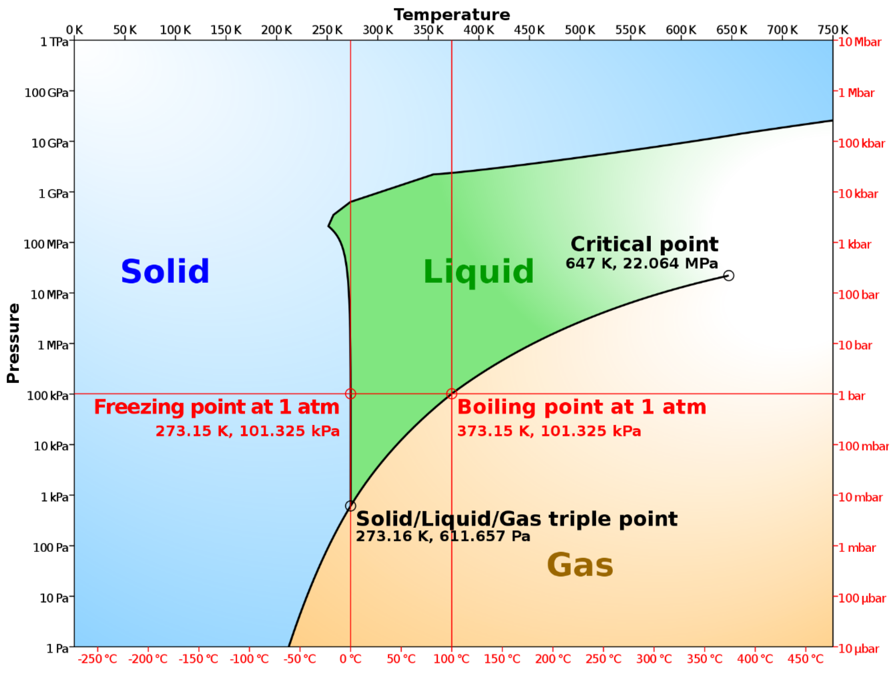

Figure 3.5: Phase diagram of water

A simple illustration of that would be the phase diagram of water: the state-space (P, T) of the pressure and temperature variables of a stable quantity of water in a closed space is generally separated into three areas depending on the physical state of the water at the different (P, T) valuations. This division of the state space of the system *water* is then based on an intrinsic property of the system and is illustrated in Figure 3.5, which is the classic representation of this system.

This figure highlights the classical knowledge that at a pressure of 1 atm, the phase of water (and therefore the qualitative state of the system) will change at $T = 273.15$ K and $T = 373.15$ K.

More generally, this diagram highlights that any discretization implies the apparition of frontiers between the designed qualitative states of the model.

In the phase diagram, the frontiers are placed to follow the change of state of water in the state space containing the variables of temperature and pressure. If we consider that water can only be in one single phase at a time (which is technically wrong but can be considered an acceptable simplification), this results in a partition of (T, P) that is commonly used by many scientists to represent the behavior of nature and reason on it.

Qualitative Differential Equations and Qualitative Simulation

A basic choice to represent the evolution of a set of continuous variables X during a time period is to write them using differential equations or systems. In its general form, a differential equation relates a variable or a set of variables to its derivatives of different order. In our works, we focused on the specific case of (*ODE*)s, which corresponds to more restrictive systems but which are far more convenient and widely used in various scientific domains.

Definition 26 (Ordinary Differential Equation) An *ODE* is a differential equation relating a variable X to its successive derivatives and which only depends on one exogenous variable y. If we note $X^{(k)}$ to be the k^{th} derivative of X according to the variable y, a n^{th} order *ODE* is a relation of the form $F(X, X^{(1)}, ..., X^{(n)}, y) = 0$. To represent the evolution of X during time, the variable y is often chosen to be the time parameter t.

Remark 3 *It is important to note that by concatenating X and its derivatives in one vector of variable, any n^{th} order ODE can be converted in a first order ODE of the form $\dot{X} = F(X, t)$.*

The *ODE*s are here chosen rather than partial differential equations because they can easily be used to represent the evolution of the system variables during time, which is often the considered model in CPS. *ODE*s are vastly used to represent various models with connected variables evolving continuously with time [103]. Moreover, *ODE*s are generally more suited to allow classic *SMT* solvers to highlight a solution to the given problem and to get a usable formula for the continuation of the abstraction process. A differential equation $\dot{X} = F(X, t)$ associated with an initial condition X_0 is called a Cauchy problem.

Different hypotheses can be made on the properties of the function F depending on the requirements of the equations. In our case, we will suppose that F is continuous and differentiable on the operational domain of X noted $\mathbf{X}$.

In general, such representation of a system allows the execution of numerical simulation that can compute with great precision the different trajectories of the system depending on its initial conditions, the execution time, and the time step.

From this well-established base of differential equations, different works have been proposed to adapt the concept of dynamic representation to qualitative reasoning.

Definition 27 (Qualitative Differential Equations) A qualitative differential equation (*QDE*) [87, 69] is the abstraction of an *ODE* where the operators between the different terms are replaced by qualitative constraints. Such an equation consists of constraints on the sign of the term $F(X, t)$ and replaces the traditional operators and functions with abstract operators such as $ADD, MULT, M^+, M^{++}$ to represent the relations between the terms.

The abstract operators, defined in [89], represent the high-level functional relation between different variables. They associate the dynamics of a set of variables to a set of others. For example, $x = M^+y$ means that x is an increasing function of y and that their dynamics are therefore related (M^{++} would imply x to be a strictly increasing function of y). $x = ADD(y, z)$ is equivalent to say that $x = y + z$.

In a *QDE*, the variable terms are related to qualities rather than quantities, but variables are still described using their relations and previous values as in an *ODE*. The abstraction of the operators and operations of the *ODE* to create the corresponding *QDE* implies that one single *QDE* may correspond to many different *ODE*.

Let there be F an *ODE* and F_Q the corresponding *QDE*. If F admits a solution f, then f is included in the solution f_q of F_Q in the sense that f_q is an abstraction of f [92].

Moreover, the solution of a *QDE* is given as a succession of qualitative states and directions on rectangles rather than a classic numerical expression.

Remark 4 *If it is possible to write a Cauchy problem from an ODE and an initial constraint, we can also write a qualitative Cauchy problem using a QDE associated with a qualitative initial constraint.*

Example 8 *Let us consider an ODE of the form* $\dot{x} = 3x + y^2 - 1$. *This equation can be abstracted in a QDE using the expression* $\dot{x} = ADD(x, M^+(y))$.

Remark 5 *As for the abstraction of a continuous state space, there does not exist a unique and universal abstraction for each ODE. Any ODE can be associated with multiple QDEs depending on the abstraction criteria and the modeling choices.*

Using this abstraction of differential equations, Kuipers has proved in [69] that it is possible to perform a qualitative simulation of a system that can compute a set of qualitative behaviors from its model and an initial qualitative state.

Qualitative simulation [104] allows the creation of a qualitative state space and a qualitative trajectory from the *QDE* by propagating the initial state. It will represent the state space of each of the considered variables. The definition of such a discretization requires the choice of **landmarks** that split the continuous state space into subsets associated with labels and representing the system's qualities.

The qualitative simulation computes the system behavior as a tree structure, given that the same qualitative state may generate more than one successor following the same *QDE*. For each branch of the tree, the simulation continues until a stable state, an already explored one, or a virtual state only reachable when $t \to \infty$. In the case where more than one successor is technically possible, the simulation splits the current branch into the same number of different sub-branches in order to represent all the possibilities. It is to be noted that this case represents one of the weaknesses of qualitative simulation: the abstraction process reduces the precision of the computation, and the non-deterministic situations may appear more often than desired.

Once the complete behavioral tree is computed, it is necessary to prune it as many trajectories among the obtained behavior are actually fictive and do not correspond to any concrete behavior [87].

As the set of qualitative states is usually finished, the algorithm has, at most, the complexity of a tree search among a finite set of nodes and is, therefore, guaranteed to terminate in this situation.

Qualitative simulation has been implemented in many tools, and the most significant is *QSIM*, developed by Kuipers and described in [69]. The *QSIM* algorithm can be described as follow:

- Create the initial qualitative states Q_i from the numerical constraints on the variables.
- Create a tree structure and place the initial states at the root and in a frontier list.
- While the frontier is not empty : take $Q_s \in$Frontier.
- For each variable of the system, use the continuity and transition constraints to determine which are the possible successors of the current state.
- For each possible successor, compute the satisfiability of the state constraints and the global constraints of the system (i.e., the invariant constraints). The successors that do not respect the invariant are suppressed.
- For each correct successor, check if the state has already been visited or if it is a state only reachable when $t \to \infty$. If not, add it to the frontier list and in the tree structure as a successor of the current state.

Qualitative simulation can be used for model checking [69], for test design and execution [92], or for exhaustive state space exploration.

The result of this simulation is a tree of qualitative trajectories with the initial qualitative state at its root and highlighting every possible qualitative trajectory from this point.

Example 9 *Let us consider a system of a tank with two holes that we fill with water, as represented on Figure 3.6. This example is inspired from the case studies presented in [69] and in [87]. Water fills the tank from the upper pipe with a constant flow f_i. Once in the tank, water gets out by the hole in the bottom with a maximum flow f_{o1}. If the water level h_w gets above the height h_{o2}, the water flows by the opening on the right with a maximum flow f_{o2}. Finally, if h_w reaches the height of the tank represented by h_m, the tank overflows. The considered variables will be h_w, f_{o1}, and f_{o2}. To make it simple, consider that the system begins when $h_w = f_{o1} = f_{o2} = 0$.*

The initial state of the system (characterizing the qualitative state and direction of the level of water) is, therefore, $QS_0 =< 0, 0 >$, meaning that we begin with an empty tank with an incoming flow equal to zero. Supposing then that the incoming flow changes and is fixed to a positive value f_i>0. Then, at this precise moment, the qualitative state of the system becomes $QS_1 =< 0, + >$, which corresponds to a tank still empty but whose water level h_w is about to increase. Then, for the successors of QS_1, their value will depend on the maximum outgoing flow f_{o1} compared to f_i. This dependency creates a division in the behavior tree and two different branches. On the first branch, if $f_{o1} \geq f_i$, the water level will stay at zero, and the initial increasing dynamics will disappear. Therefore, the system will stay on the stable state $QS_{f1} =< 0, 0 >$. Otherwise, the system will at first fill faster than it empties. In this case, the successor state of QS_1 will be $QS_2 =< (0, h_{o2}), + >$. After having sufficiently increased, h_w will reach the level of the second outgoing pipe. At this instant, the qualitative state will be $QS_3 =< h_{o2}, + >$.Then, once again, the qualitative behavior tree will split into two branches whose possibilities will depend on the order relation between f_i and the maximum outgoing flow $f_{o1} + f_{o2}$. If $f_{o1} + f_{o2} \geq f_i$, the system will stay in a terminal state $QS_{f2} =< h_{o2}, 0 >$. Otherwise, the water level will still increase, and the associated qualitative state is $QS_4 =< (h_{o2}, h_m), + >$. Finally, on this last branch, the system will reach its maximum filling rate in the qualitative state $QS_5 =< h_m, + >$and the water tank will overflow, producing the qualitative state $QS_{f3} =< h_m, 0 >$.

The obtained behavior tree is represented in Figure 3.7.

Example 10 *Let us consider a system involving a temperature θ following the differential equation $\dot{\theta} = f(\theta, t)$ and varying on $\mathbb{R}^+$. It is possible to divide $\mathbb{R}^+$ in different open intervals such as $(0, 273.15), (273.15, 373.15), (373.15, +\infty)$ separated by landmarks on the points $\{273.15, 373.15\}$ and labeled with the qualities $\{$cold, medium, hot. Then, the same abstraction can be done on $\mathbb{R}$ the definition space of $\dot{\theta}$, in the intervals $(-\infty, -100)$, $(-100, -10)$, $(-10, 0)$, $(0, 10)$, $(10, 100)$, $(100, +\infty)$ using landmarks on the points$\{-100, -10, 0, 10, 100\}$ and associated with the labels {decreasing fast, decreasing, decreasing slowly, constant, increasing slowly, increasing, increasing fast.*

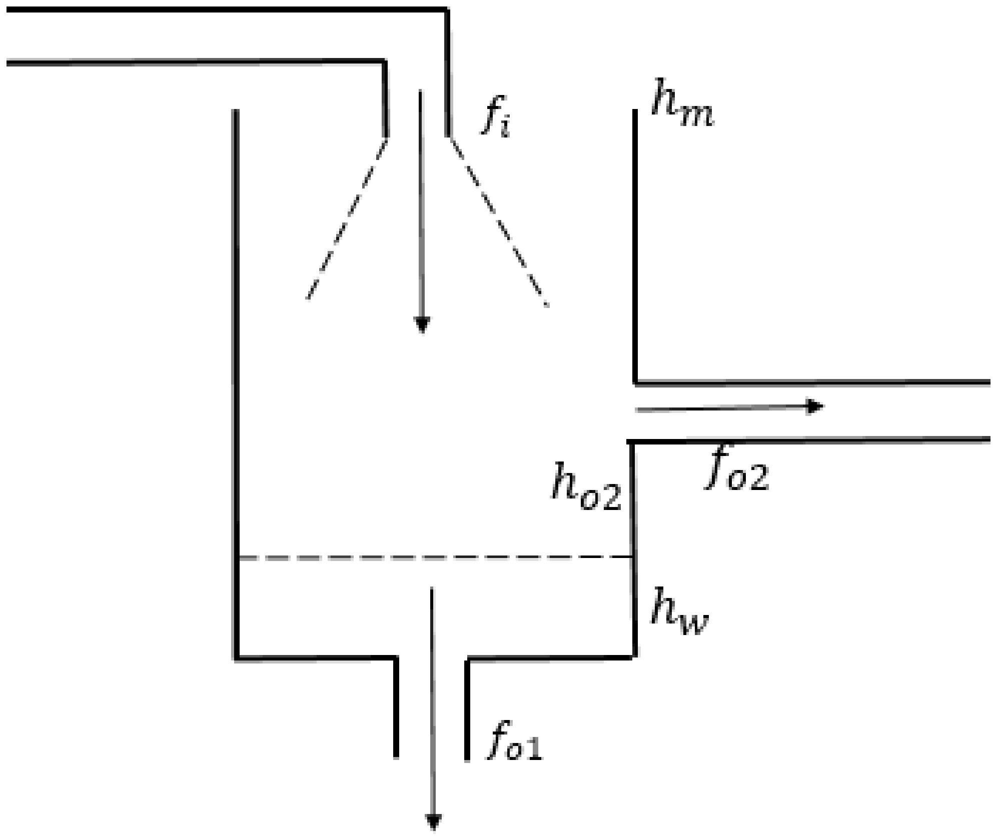

Figure 3.6: Representation of the tank with two openings

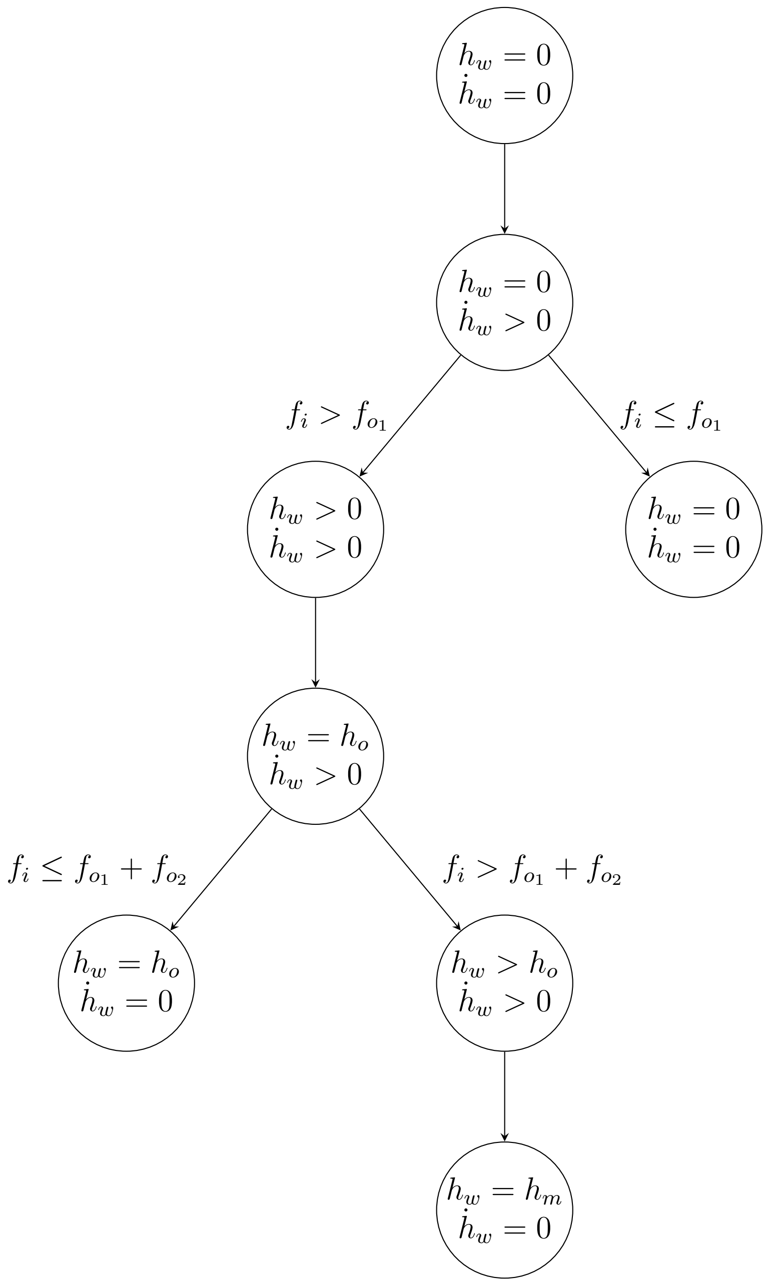
$h_w = 0$ $\dot{h}_w = 0$
$h_w = 0$ $\dot{h}_w > 0$
$f_i > f_{o_1}$
$f_i \leq f_{o_1}$
$h_w > 0$ $\dot{h}_w > 0$
$h_w = 0$ $\dot{h}_w = 0$
$h_w = h_o$ $\dot{h}_w > 0$
$f_i \leq f_{o_1} + f_{o_2}$
$f_i > f_{o_1} + f_{o_2}$
$h_w = h_o$ $\dot{h}_w = 0$
$h_w > h_o$ $\dot{h}_w > 0$
$h_w = h_m$ $\dot{h}_w = 0$

Figure 3.7: Behaviors of the Tank with two Openings

Naming Q_θ the partition of the state space of θ (composed of both the intervals and the landmarks) and $Q_{\dot{\theta}}$ the partition associated with $\dot{\theta}$, the qualitative differential equation corresponding to the ODE of θ is a mapping $\delta_Q : Q_\theta \mapsto Q_{\dot{\theta}}$.

Remark 6 *If the intervals associated with their labels are considered qualitative states in a major part of the literature, there is no consensus about the landmarks. In some contributions [34], the considered intervals are semi-open, and the landmarks are, therefore, integrated into one of the considered intervals. In others [87], landmarks are considered qualitative states on their own. We favored the second option for consistency and ease of computation.*

Using the chosen discretization of the state space of the system variables and their derivatives *QDE* can be modeled using automata.

3.5 . Semi-Qualitative Modeling

Qualitative modeling aims to highlight specific features of a CPS or its behavior by sacrificing numerical precision and access to other features. Among the lost capabilities of the model is the possibility of expressing an error on the current value directly related to the availability of numerical values. Actually, expressing an error on a qualitative value or a reliability value in $\%$ is impossible with qualities rather than quantities. In order to limit this drawback or to combine the advantages of the different reasoning paradigms, some contributions developed the concept of semi-qualitative modeling [88].

3.5.1 . Interval arithmetic based solvers

Definition 28 (Interval) Let us consider $(a, b) \in \mathbb{R}^2$ such that $a \leq b$. The closed interval $[a, b]$ represent all the values c of $\mathbb{R}$ such that $a \leq c \leq b$, while the open interval (a, b) includes all the values $d \in \mathbb{R}$ such that $a < c < b$.

The first technique used to improve qualitative simulation with more numerical knowledge is based on the use of interval arithmetic [105]. Interval arithmetic extends the classic arithmetic operation on numbers to the computation of intervals, allowing to make operations from $\{+, -, \times, /\}$ between two intervals defined on $\mathbb{R}$. By applying this algebra, it is possible to use interval propagation and apply operations on intervals rather than on numerical values. The following paragraphs present some tools integrating this functionality and aiming at improving the results of qualitative simulation with elements from interval propagation computation.

Q2

One of the first tools developed for this purpose is *Q2*, presented in [106]. *Q2* is an extension of *QSIM* thought to associate numerical knowledge to qualitative states. Qualitative values and directions can be associated with numerical intervals, while qualitative evolution and constraints can be given numerically computable envelope functions. QSIM algorithm is then combined with an interval propagation algorithm to discriminate the possible behaviors among the QSIM-computed traces. Computing the intersection between the obtained intervals and the constraints implied by QSIM states gives a more precise constraint on the value using both qualitative and quantitative expressions.

Using Taylor-Lagrange approximation, it is even possible to compute numerical values of the variables at a given time t.

Behaviors selected by *Q2* are more precise than those highlighted by *QSIM* but are less likely to give more qualitatively understandable properties. Sometimes, the intersection between the qualitative constraints and propagated interval can be empty, meaning that the behavior does not actually exist.

Q3

Q3 is an extension of *Q2* presented in [107]. It improves the efficiency of *Q2* by adding intermediate qualitative states between the states computed by *QSIM*. As in numerical simulations, adding more sampling points to a process improves the precision of the simulation/computation. To create new states, *Q3* must also introduce new constraints to discriminate to sub-states of what would have been the same qualitative state according to *QSIM*. [107] mentions this procedure as a hybrid simulation, implying elements from both quantitative and qualitative paradigms. *Q3* is also said to be convergent as the precision of the result increases with the reduction of the simulation step.

SQSIM

Finally, *SQSIM* [108] has been developed as a combination of the previously mentioned solvers and other more numerical tools also using interval propagation methods. Using the results obtained by the different solvers, *SQSIM* computes the intersection of the different intervals to give a more precise and restricted uncertainty envelope. However, it is entirely dependent on the reliability of at least three different tools, which must all give coherent results. If the intersection between the different trajectories is empty, *SQSIM* will not make the effort to compute the most likely and will just consider the trajectory not to exist.

3.5.2 . Fuzzy-logic based solvers

Other extensions of *QSIM* like *FuSim* [109] instead use fuzzy logic paradigm [110] in order to better characterize the qualitative states obtained by *QSIM*. Fuzzy logic has been conceived to deal with imprecise/unknown values, using multivariate and probabilistic logical systems containing more than the classic $\{True,\ False\}$ boolean values. Actually, fuzzy logic uses a set of values between $True$ and $False$, characterizing both the absence of certitude about the real boolean value and containing tendencies about the one more likely to correspond to reality.

3.6 . Order of Magnitude Reasoning

Among the various tools used to reason on qualitative models, we can find the orders of magnitude [111]. This method expands the reasoning on sign, which is the very base of qualitative methods [66, 69]. It is mainly supposed to avoid the limitations of the sign algebra, especially the difficulty of applying the $+$ (and less significantly $\times$) operators to values of unknown magnitude. Taking the magnitude of a value into account allows for more precise and coherent computations but also has drawbacks on the permissiveness of operators [112]. For example, transitivity, associativity, and distributive properties are more limited in this qualitative algebra. Order of Magnitude Reasoning (OMR) can be absolute (comparison to fixed values) or relative (comparison between variables) [113].

The first option to reason about orders of magnitude implies partitioning the state space according to constant landmarks. Variables are individually compared to these reference values k_i and placed in the corresponding rectangle (multi-dimensional closed interval), determining their magnitude. It also requires choosing a scale to apply to these reference values that will set the abstraction's regularity and granularity. The interest is that by optimizing the choice of landmarks, it is possible to abstract the precise values completely while preserving helpful information about the value of the state variables. The scale can, for example, be linear or, if the variable is to stay positive, logarithmic and rely on powers of k_i. The values k_i will here depend on the design criteria of the system, such as the initial value or the anticipated extreme points. When the sign can change, a linear scale may be preferred. More generally, the choice of the state space partition will strongly depend on the simulation's objectives and the system's design [114]. For example, a linear scale in one dimension based on a unique value k would be structured with landmarks on the values $0, k, -k, 2k, -2k, \ldots$. It could be used in the case of the model of the thermostat, where the unique system variable varies between values that are simultaneously closely spaced and sufficiently far from zero to make any logarithmic scale unusable. In this case, an adapted scale could be a linear one with a value $k = 2$.

A second option is to use relative orders of magnitude. This option compares variables with each other rather than with reference values. This requires the physical quantities and units to be comparable. Different comparison systems exist, such as **FOG** [113], **O(M)** [111], or **Rom** [115]. These systems do not use the same operators, but some of them are used in most systems: Ne (*negligible*, also noted $\ll$), Co (*comparable*), and Vo (*neighbor*, or very close to). These operators can give more precision than the usual ones, such as $=$, $<$, $>$, or $\approx$. Orders of magnitude are mainly used in purely qualitative algebra because the intrinsic properties are incompatible with the archimedean property and, therefore, not supported in $\mathbb{R}$. For example, it is supposed in **Rom** that if $a\,Ne\,b$, therefore $(a+a)\,Ne\,b$, which is not correct in archimedean spaces. Some recent works added the operator Di, meaning *distant* [87], and corresponding to the situation where the ratio between two values is too important to consider them as comparable (not of the same order of magnitude), but too small for one of them to be negligible. Absolute and relative orders can also be combined, but this requires an adapted space partition and a precise definition of the relative operators. The difficulty is to make the different frontiers match each other. Otherwise, we get the superposition of two independent measure scales, adding too much complexity for few benefits. These techniques were designed to compare values, but we must also compare functions to compare the behaviors of variables and not only static values.

3.7 . Order of Growth Reasoning

Many works of the previous century have studied categories of functions, and some categorized them using their order of growth (OG). This concept eases the representation of the behavior of a function $f : t \mapsto f(t)$ when $t \to \infty$. The notion of OG was first formalized by Borel [116] and pushed further by other scientists using the works of Hardy [117] on logarithmic-exponential functions (set of functions obtained by addition, multiplication, or composition of logarithm or exponential functions, closed under addition, multiplication, and composition). Let us consider $f : \mathbb{R} \mapsto \mathbb{R}$ a function. We define the OG of f by

$$c(f) = \lim_{t\to+\infty} \frac{ln|f(t)|}{ln(t)}$$

and $c(f : t \mapsto 0) = -\infty$. In the case of polynomial functions $f : t \mapsto \sum_{i=0}^{n} a_i t^i$ with $a_n \neq 0$, we have $c(f) = n$. When f is the sum of many terms, $c(f)$ only considers its asymptotic dominant term without consideration of the others. Moreover, $c(f)$ only has a meaning if f converges or diverges to an infinite: if f is periodic or has no clear limit when $t \to \infty$, then $c(f)$ does not exist. The expression of c allows us to highlight some properties of the OG regarding its reaction to operations. Using the properties of the logarithm, it appears that for two logarithmic-exponential functions f and g, $c(fg) = c(f) + c(g)$ and $c(f \circ g) = c(f)c(g)$. Moreover, if $c(f) \neq c(g)$, then $c(f+g) = max(c(f), c(g))$.

Still, this measure of asymptotic behaviors has some weaknesses. First, with the given formula, it is impossible to distinguish constant functions from logarithmic ones or to make a difference between two exponential functions with their OG. Actually, $c(f \mapsto k \neq 0) = c(f : t \mapsto ln(t)) = 0$, and also $c(f : t \mapsto e^t) = c(f : t \mapsto 10^{2t}) = \infty$, which is a severe limitation for a behavior classification. Classifying an infinite-diverging and a constant function with the same value cannot be sufficient as a characterization. Finally, OG computation only applies when $t \to \infty$, making it useless for analyzing systems at finite time scales through simulation.

3.8 . Qualitative Models for Unspecified Dynamics Systems

In some partially known systems or in early design phases of CPS, it may appear that the dynamics cannot be expressed using traditional differential equations. In order to express the known dynamics relations, some contributions [68, 97] developed an execution language to qualitatively link the value of the variable derivative with respect to time with others variables or parameters of the system. In [68], the authors specifically develop relations based on causality reasoning with four operators $PROP$, $CPROP$, $IPROP$, and $CIPROP$.

The common $PROP$ string characterizes a qualitative proportionality between the related elements (i.e., an imposed similarity of their signs).

$IPROP$ designs an inverse qualitative proportionality, meaning that the related variables have an opposite sign.

Finally, the letter C stands for "causal", in the sense of causal ordering. More precisely, it implies that the sign dependence only works in one sense.

For example, if $a\ PROP\ b$, any change of sign of any of the two variables a and b will immediately cause the other to follow it. On the contrary, if $a\ CPROP\ b$, a change in the sign of a will not influence b, while a change in the sign of b will cause a to change its sign in the same direction.

This difference converges with the ideas of different orders of dynamics relations developed by Mosterman in [94] to diagnose hybrid systems.

Symmetric proportionality corresponds to static relations, while causal links can be compared to dynamic relations (where variables depend on the derivatives of others), which are asymmetric and longer to observe.

This language can be used to model the qualitative evolution of the different variables of a system with very little knowledge about their dynamics; the quality of the obtained information at the end of the propagation is not sufficient to fully explore the state space of the system and to discriminate between different qualitative behaviors.

Moreover, the absence of any consideration of magnitude or time value makes it impossible to deal with any duration in any qualitative state. Therefore, it cannot be considered as a solution whenever a symbolic expression of the differential equations of the dynamics is available.

3.9 . Time Management and Qualitative State Duration

In regular hybrid automata, the time elapsed in any operating mode or qualitative state can be achieved using a stopwatch (i.e., a clock variable whose flow condition is defined as $\dot{x} = 1$ if the system is in the mode m or in the state q, and $\dot{x} = 0$ otherwise). Therefore, a simple subtraction reveals the time elapsed in the researched area.

One of the drawbacks of qualitative simulation tools like QSIM is that time is represented symbolically, and it is, therefore, not possible to compute the time passed in each state/mode. Therefore, the previous method cannot work anymore in this situation.

One of the methods is to use temporized automata with clock variables. However, some contributions, such as [118], introduced techniques to make time information numerically explicit. In this example, the authors used the Taylor formula to express the variation between two time steps with the initial and final values and with the elapsed time. The duration dt elapsed between two landmarks x_i and x_j can be written

$$dt \approx \frac{x_j - x_i}{qdir(x)}$$

with $\approx$ denoting qualitative equality. As qualitative states are characterized by qualitative position and directions, $qdir(x)$ is the qualitative direction of the variable x in the qualitative state between x_i and x_j, i.e., the qualitative abstraction of $\dot{x}$. A problem posed by our situation is that we mainly deal with qualitative states defined by equations that are more complex than landmarks. This time computation formula indeed does not fit qualitative states whose limits are nullclines defined by equations such as $\dot{X}_i = 0$ with $i \in [\![1, n]\!]$.

In the case of timeless automaton or timeless abstraction of timed systems, the computation of a duration d in a qualitative state is much more complex.

[119] expresses the need to use complex metrics (they introduce a relative space metric to measure the time elapsed) in order to compute the time duration of operating modes because the measurement of distance using Euclidean measure may not allow the abstraction of a system to a temporized model.

Temporizing a timeless system and measuring time in a mode or in a qualitative state is a problem we did not consider, but that should be solved to overcome the challenges posed by the most complex systems.

4 - SYSTEM ABSTRACTION

Note: This chapter is a reproduction authorized by Springer of a published article in ISSE [120].

In this chapter, we expose the abstraction process that we constructed to create a qualitative model from the input CPS and compute the qualitative trajectories corresponding to its dynamics. Section 4.1 details the state space abstraction process, which creates a finite partition of the system state space fitting the constraints of qualitative reasoning. In section 4.2, we explain the process employed to compute a qualitative abstraction of the system behavior on the chosen partition based on its dynamics equations. Finally, section 4.3 presents the difficulties raised by the most complex systems that imply a large number of variables and how they can be solved.

4.1 . State Space Discretization

Definition 29 (Discretization) If $\mathbb{K}$ is a continuous set, we call discretization of $\mathbb{K}$ a finite partition of $\mathbb{K}$ in $\mathbb{K}_i \subset \mathbb{K}$ with $i \in [\![1, k]\!]$ and $k \in \mathbb{N}$, meaning that $\bigcup_{i\in[\![1,k]\!]} \mathbb{K}_i = \mathbb{K}$ and $\forall\, (i, j) \in [\![1, k]\!]^2, i \neq j \implies \mathbb{K}_i \cap \mathbb{K}_j = \varnothing$.

The first step in computing the qualitative trace of a system is to transform our representation of the system to a qualitative model to serve as computation support.

Definition 30 (Qualitative Model) Let S be either a CPS or a numerical representation of a CPS. A qualitative model associated with S is a representation of S trading precision and knowledge for a convenient highlight of the relations between the elements of a discretization of the system. These relations can be a causal inference or a temporal ordering.

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 *ODE* $\dot{X} = F(X(t), t)$ with t the time parameter of the system.

Example 11 *To illustrate our explanations, we will take as an example the hybrid Brusselator system, expressed as:*

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

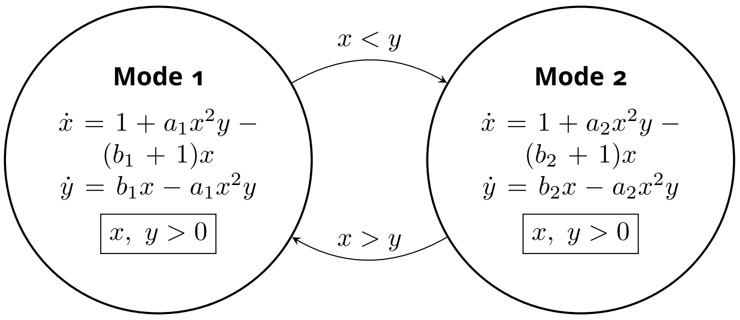
$x < y$
Mode 1
$\dot{x} = 1 + a_1x^2y - (b_1 + 1)x$
$\dot{y} = b_1x - a_1x^2y$
$x,\ y > 0$
Mode 2
$\dot{x} = 1 + a_2x^2y - (b_2 + 1)x$
$\dot{y} = b_2x - a_2x^2y$
$x,\ y > 0$
$x > y$

Figure 4.1: Hybrid model of a Brusselator system

With this representation, it is possible to visualize the discrete part of the model as a hybrid automaton (see Figure 4.1). Different abstraction methods have been introduced and used for continuous evolution. The choice of the abstraction method is critical because it will strongly influence state space exploration. Among the methods developed and studied, the most noticeable for CPS study 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 state space is made separately for the different variables of X and for their derivatives. The abstraction of the state space is processed using landmarks on the zeros of the variables and abstractions of the system's differential equations that are the QDE. These abstractions allow any change of sign of a component X_i to influence the other components. 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 gives little 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, suppose 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. In that case, they will be considered as reference landmarks to compare and abstract the current value of the speed. This integrates elements of OMR [87]. However, as these values are not from the sign algebra, they cannot 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 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 contradicts 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 [35] 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. Tiwari assumes that all these equations have a polynomial form according to the components of 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 the 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. One can note that previously mentioned landmarks can be integrated into these polynomial equations once represented as $X_i - c_i$ for a landmark $X_i = c_i$. 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. This condition quickly suppresses the nilpotent and idempotent polynomials. As the considered 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 $\dot{x} = 1 + ax^2y - (b+1)x \in \mathbb{K}[x, y]$ 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 according to the time parameter creates a refinement of the qualitative model. However, it may increase computational complexity up to a certain point 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 corresponds to the search for a trade between precision and complexity. This criterion must be chosen before the execution, so there is still a problem with non-instantiated systems. If no information is available about the needed precision of the abstraction or the usefulness of new reference values, the criteria will have to be chosen arbitrarily. In [35], the author did not consider this a problem, as whenever the discretization is stopped, the result is still an abstraction of the system. He does not see over-refinement to be disturbing. However, depending on the use of the model, the searched precision will be completely different.

The search for stopping criteria for the abstraction process is a challenge on its own and would require further investigations. We do not see the expression of a universal criterion as possible from what emerged from the current works. However, something quite general can be proposed. The first criterion we could add to the modeling process is a constraint on the maximal complexity of the resulting qualitative model. This complexity can be expressed with the number of elements of P_m, giving a maximum estimation of the possible number of existing qualitative states and, by a simple computation, the maximum estimation for the number of transitions. This computation of the complexity of the model is an analogy of state space complexity [121] to the discretization of the system state space. For a discretization achieved with a set P_m of size l_m, we can bound the maximum number of qualitative states in the mode m to 3^{l_m}, and therefore the maximum number of transitions between these states to 3^{2l_m}. The maximum complexity of the qualitative model is, in consequence, directly dependent on $|P_m|$. An idea would be to stop the abstraction process when each P_m reaches a limit size. Still, a new challenge is raised by the hybrid nature of treated systems. Should the different modes be treated separately with a maximum complexity associated with each, or should the model be treated as a whole with a global maximum complexity? We still have to study this question more profoundly. Anyway, the intuitive answer is that the consideration of the complexity of the complete model could provoke a disturbing imbalance between the granularity of the state spaces of the different modes. Therefore, the choice of a maximal authorized complexity per mode, equivalent to a maximal number of qualitative states for each mode, seems to be the most adapted solution. It would also be possible to evaluate the complexity of the abstraction and to refine in consequence not only considering the number of polynomials in P_m and their supposed inherited states and transitions but rather on the actual qualitative states and the existing transitions between them. This would require complete execution of the further process before coming back to possibly refine or reduce the qualitative abstraction of the state space. The computation of the transitions would then have to be achieved for each execution, as every refinement of the qualitative state may affect every formerly computed qualitative transition. Yet, as the creation of the qualitative model is supposed to be computed offline without severe time constraints, this possibility is not to be rejected.

In contrast, another option to define a stopping criterion is to evaluate the quality of the obtained discretization. This is equivalent to an optimization exercise. It requires the definition of a utility function f_u to evaluate the quality of a given abstraction. Precision cannot be sufficient to constitute a criterion, as qualitative reasoning implies inherently a loss of precision. Again, defining a utility function pertinent to every qualitative abstraction seems impossible, but some specific points could be considered to develop a generic optimization process. The main quality intended for a qualitative model is the ability to correctly reason on qualitative values to get satisfying results without needing numeric computation. The criterion to determine a sufficiently precise abstraction could be the ability to discriminate a given set of numerical values in the same number of different qualitative states. If two of the given values were to be abstracted in the same qualitative state, the abstraction would be considered insufficiently refined, and the computation of P_m would restart to generate more qualitative frontiers and, therefore, more qualitative states discretizing the state space.

The last challenge regarding the management of P_m and the discretization of $\mathbf{X}$ for each mode once the stopping criteria are chosen is the priority order to attribute to each element of P_m. Should the same element be derived many times in a row due to its important role in the description of the system, or should there be no priority between the considered polynomials? As the flow equations and their derivative contain information about the system's dynamics, giving them higher priority than the derivatives of the transition condition would make sense. This decision is crucial to drawing the abstraction policy and must be made according to the desired qualitative information to integrate into the qualitative model.

For the rest of the thesis, the stopping criterion was chosen as simple as possible to allow the algorithm to terminate without risking influencing the results in an undesired way. We fixed for each of the polynomial equations of the system corresponding to the initial elements of P_m a maximum number of times to be derived during the process before stopping the P_m computation.

Finally, in order not to integrate the same polynomial p more than once in the same P_m, it is necessary to add a unicity test during this discretization phase.

In the hybrid Brusselator system, with a simple criterion of a maximum of 2 derivations, 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 added.

Once P_m is computed, the next step is to take every $p \in P_m$ and use a polynomial solver to solve the equation $p = 0$. The obtained solutions give the expression of the nullclines (i.e., a curve supported by an equation $\dot{v} = 0$ with v a function of time) of the dynamics and allow a discretization of the state space of the variables. Each value of X will now be abstracted by comparison to these nullclines.

Proposition 1 *Using the so-obtained discretization of the state space of each mode of a CPS, each valuation x of X can be abstracted in exactly one qualitative state.*

This proposition comes from the nature of discretization, which is a partitioning operation of the state space.

Definition 31 (Qualitative State) We call **qualitative state** of a system the pair (m, qs) with $m \in \mathbf{Q}$ 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. A qualitative state corresponds to a set of abstracted values of X expressed as a vector of elements from $\{+, -, 0\}$ representing respectively for every $p \in P_m$ the fact that $p(X)$ is negative, zero, or positive.

A qualitative state considers not only the qualitative abstraction of the numerical value of the variable at a given time but also the abstraction of all the pseudo-variables corresponding to the computed polynomials of P_m. This includes information about the derivatives until a given order (including its qualitative direction), the relative position of the variable to given frontiers, and the sign of specific quantities that have some importance in the model. This increases the precision and the accuracy of the computation of the qualitative behavior [122].

Definition 32 (Abstraction Function) Given a CPS S defined with the previous notations, we define the system abstraction function

$$A : \begin{cases} \mathbf{Q} \rightarrow \mathcal{B}_{\mathbf{X}} \\ m \mapsto \mathbb{S}_{m,\mathbf{X}} \end{cases} \tag{4.1}$$

with $\mathbb{S}_{m,\mathbf{X}}$ the set of qualitative states defined on $\mathbf{X}$ for the mode m and $\mathcal{B}_{\mathbf{X}}$ the set of all partitions of the space $\mathbf{X}$ inspired from the notation of the Bell number $\mathcal{B}_{|\mathbf{X}|}$.

From this definition and the definition 31 of the qualitative state, we deduce the state abstraction function in the mode m noted

$$\alpha_m : \begin{cases} \mathbf{X} \rightarrow \mathbb{S}_{m,\mathbf{X}} \\ x \mapsto qs \in \{-, 0, +\}^{|P_m|} \end{cases} \tag{4.2}$$

which abstracts each value x of X as a vector of length $|P_m|$ representing the sign of each polynomial p of P_m for $X = x$.

A qualitative state represents an abstraction of a whole set of numerical values of X, therefore implying a loss of knowledge in the representation. For a qualitative state qs corresponding to the abstraction of two different numerical values x_1 and x_2, coming back from the qualitative knowledge to numerical representation implies the implementation of a concretization function [35], which cannot give with complete certainty one value rather than the other. Instead, it will return a continuous set of values allowed for the variables. As the qualitative states are computed by partitioning the state space of S for each mode m, the mutual exclusion principle automatically applies to them by definition of a partition.

The advantage of this method is that it is applicable on non-instantiated systems: it does not require prior knowledge about the context or the system's objective, meaning it is convenient to generalize.

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 [123], 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. In the case of explicit equations with non-valuated constants, α can be defined but will not be reliable before a complete definition of all the symbolic constants as its return value depends on it.

4.1.1 . Evolution abstraction

Once the system's state space is discretized in qualitative states, the qualitative model consists of a finite partition of its state space in qualitative states for each mode. Upgrading it and making it fit for complex applications such as prediction or diagnosis requires computing the possible evolution for each qualitative state, given the dynamics of the system.

Definition 33 (Continuous Evolution) Let t be the time parameter of the system S. A continuous evolution of S is a function $\sigma : t \mapsto \mathbf{X}$ that maps every defined instant with a value of X.

The stake of qualitatively studying a system relies not only on the ability to create and manipulate an abstraction of its state space. A major challenge of qualitative reasoning consists of applying this qualitative state abstraction to the system's dynamics to observe a concise view of its evolution. The continuous evolution of a system is often observed to verify specific properties and to visualize the behavior of a concrete or a well-designed system. However, when the design process is not over, numerical resolution and computation do not fit the available knowledge nor the qualitative model previously created.

To create a behavior representation fitting qualitative reasoning, the study of the dynamics must follow the mindset of qualitative modeling and create an abstraction of the continuous evolution σ. Therefore, we will consider a decomposition of the system evolution inspired from [66] using events and tendencies.

Definition 34 (Events and Tendencies) Let us consider a continuous evolution σ of the system S on the time interval T. We note m_t the operating mode of S at each time t. We consider as events the values of t corresponding to valuations x of X associated to a change of qualitative state of S, i.e., a time t such that $S(t) = (m_t, x)$, $\exists p \in P_{m_t}$ such that $p(x) = 0$, and such that $\exists \epsilon > 0, \forall \theta \in (0, \epsilon), S(t \pm \theta) = (m_2, x_2) \neq (m_t, x)$. On the opposite, we consider as tendency an interval $T_i \in T$ such that $\forall t \in T_i$, if $S(t) = (m_t, x_t)$, then $\forall p \in P_{m_t}, p(x) \neq 0$.

For example, in a sine function $f : x \in \mathbb{R} \mapsto sin(x)$, where the function and its first order derivative are studied, the events will be the points $\{k\frac{\pi}{2}\}_{k \in \mathbb{Z}}$ because the even values of k will imply a change of sign of f while the odd values of k correspond to the critical points and to a change of sign of $\frac{df}{dx}$.

Consequently, the tendencies of f will correspond to the intervals $(\frac{k\pi}{2}, \frac{(k+1)\pi}{2})$ where neither f nor $\frac{df}{dx}$ change their sign.

Events and tendencies offer support to the abstraction of continuous trajectories, as any continuous behavior can be represented using a variation table, whose extrema and thresholds correspond to events while the tendencies represent variation directions between the events.

Definition 35 (Qualitative Evolution) Let t be the time parameter of the system S. A qualitative evolution of S is a function $\kappa : t \mapsto \mathbf{Q} * \mathbb{S}_{\mathbf{Q},\mathbf{X}}$ that maps every defined instant with a couple (m, s_q) with m the mode and s_q the qualitative state corresponding to the abstraction of the continuous value X of the state of the system at time t. Said otherwise, a qualitative evolution of a system corresponds to its continuous evolution abstracted with tendencies and events.

To abstract all the possible behaviors of a system, we must first fix and abstract its initial state to figure out which qualitative state to begin with. To this extent, it is necessary to define which mode m of Q is the initial one. We then apply the abstraction function α_m on the initial value X_0 of X. The result s_0 corresponds to the initial qualitative state of the system S. Without an initial numerical state, the initial qualitative state may be chosen arbitrarily in the authorized set of qualitative states.

From s_0, the objective is now to apply the dynamics F of S on the state space abstraction from s_0 to propagate the qualitative states and explore all the possible qualitative behaviors κ of the system.

Then, while we are not in an absorbent state or 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. $frontier$ contains the qualitative states from which the analysis should progress, and $explored$ memorizes the ones already studied.

4.1.2 . Qualitative transitions

Definition 36 (Qualitative Transition) A transition $tr = (m_i, s_1) \rightarrow (m_j, s_2)$ is called qualitative or intra-modal iff $m_i = m_j$ and $s_1 \neq s_2$. Moreover, tr must be allowed by the theorems of continuity, such as the intermediate value theorem.

While $frontier$ is not empty, we consider s_i the next state in $frontier$. From this state s_i, the objective is to compute all the successor states according to the dynamics of the current mode m of S. The first step is to compute all the states sharing a border with s_i. This is possible by using a polynomial constraint solver such as *Z3* and by translating each digit of the state-vector $v_s \in \{-, 0, +\}^{|P_m|}$ to a constraint corresponding to the associated sign and imposed to the related polynomial. Changing one digit of the current qualitative state and respecting the intermediate value theorem allows the creation of all the theoretical neighbors of s_i. However, many of the found neighbors do not exist or do not offer a transition from s_i.

Using constraint solving, we then find the possible neighbors s_j. Considering the hypothesis that every equation defining the qualitative states is polynomial, this resolution only consists of a conjunction of polynomial inequalities that can be solved using the chosen solver. Then, if s_j exists, it is placed in the structure $RealNeighbors$.

For s_k in $RealNeighbors$, we must verify whether or not a transition from s_i to s_k is possible in the behavior of the system. This is achievable by using the Lie derivative formula [35, 124]:

$$L_X(p) = \sum_{X_i \in X} \frac{\partial p}{\partial X_i} \frac{\partial X_i}{\partial t}. \tag{4.3}$$

with $p \in P_m$ the polynomial function supporting the border between the qualitative states s_i and s_k, and the X_i the components of X.

Proposition 2 *The transition from s_i to s_k is possible iff $L_X(p)(p(s_i) - p(s_k)) > 0$ where p is the polynomial function supporting the border between the qualitative states s_i and s_k.*

Proof see [35].

Using the Lie derivative, it is possible to compute every qualitative transition in a mode by applying this process to every state $s_i \in \mathbb{S}_{m,\mathbf{X}}$, and to visualize the behavior tree of any execution in a continuous model or a single mode of a hybrid model. The different tests allow the suppression of the non-existing states and the computation of only the real qualitative transitions. Using the $existing$ structure allows us to avoid the loops and suppress the quiescent, the already explored, and the terminal states. However, dealing with modal transitions of a hybrid system is more complex and requires more computation.

4.1.3 . Discrete transitions

Definition 37 (Modal Transition) A transition $tr = (m_i, s_1) \rightarrow (m_j, s_2)$ is called discrete or modal if $m_i \neq m_j$.

Computing the possible discrete transition in a qualitative hybrid model requires incorporating the guard conditions of each transition from the current mode in the abstraction process. To simplify the computation, we consider the guard condition necessary and sufficient to provoke the transition, i.e., a discrete transition happens as soon as its associated guard condition is verified. As the guard conditions are considered polynomials and are incorporated in $P_m \; \forall\, m \in \mathbf{Q}$, the verification of the guard predicates can be integrated into the qualitative transitions detection process. When a polynomial p associated with a guard condition changes its sign, the guard condition is verified, and the transition is triggered. For the involved transitions, the jump is not from a qualitative s_i to another s_j in the same mode, but between the qualitative states (m_1, s_i) and (m_2, s_k) with $i \neq j$. The intended transition is not qualitative anymore but becomes modal. Therefore, the change concerns not only the continuous variables of X but also Q. Moreover, a shift in mode implies activating an associated reset function res that applies on X and may also cause a discontinuity in its valuations.

Proposition 3 *Abstracting the discrete transition $tr = (m_i, s_1) \rightarrow (m_j, s_2)$ requires to compute $\alpha_2(s_1)$, i.e. $\alpha_2(x)$ for all $x \in s_1$.*

The computation of $\alpha_2(s_1)$ can be achieved by creating another iteration of a polynomial solver: by using branch-and-bound solving on all the constraints defining the qualitative states on m_2, computing all the possible destination states for a reset function applied on (m_1, s_i) is possible. The list of obtained authorized states $[(m_2, s_j)]_{s_j \in \mathbb{S}_{m_2, \mathbf{X}}}$ will constitute the successors of (m_1, s_i).

Doing this for all the triggering transitions in a mode m_1 highlights all m_1 outgoing transitions. Once computed for every mode $m \in \mathbf{Q}$, all the system's outgoing and incoming modal transitions will be represented.

4.1.4 . Abstraction refinement

Definition 38 (Partition refinement) Given $\mathbb{K}$ a set and $P_1, P_2 \in \mathcal{B}_{\mathbb{K}}$ two partitions of $\mathbb{K}$. P_2 is said to be a refining partition of p_1 if $\forall\, s \in P_2, \exists!\; s_1 \in P_1$ such that $s \subseteq s_1$.

As the state space abstraction to create qualitative models is based on state space partitioning, it is possible, given a first system abstraction, to generate a refined discretization of its state space to improve the knowledge included in the qualitative states. This refinement of an already-defined abstraction goes through the addition of new functional constraints in the list of polynomials or rational functions that define the first abstraction.

Definition 39 (Abstraction refinement) Given a CPS S on which we defined the abstraction functions A and A' associated with $\alpha_{m, m \in \mathbf{Q}}$ and $\alpha'_{m, m \in \mathbf{Q}}$. The abstraction of S defined by A' and $\alpha'_{m, m \in \mathbf{Q}}$ is a refinement of the first one if $\forall\, m \in \mathbf{Q}$, $A'(m)$ is a refining partition of $A(m)$.

A refinement of a qualitative model can be applied by adding new constraint equations to the list P_m for every operating mode m, which will add as many qualitative frontiers in the system state space and, therefore, increase the number of qualitative states in the state space partition.

Consequently, the qualitative and discrete transitions of a refined system will have to be recomputed, as the previous ones are between qualitative states that may have been divided.

4.2 . Qualitative Behavior Computation

By neighborhood propagation, we can compute all the paths among the defined qualitative states of the qualitative model considering its dynamics. Adjacent state propagation (see Definition 40 below) allows us to determine all possible variation directions from an initial state. The obtained set of qualitative traces contains all the theoretically possible behaviors of the system S.

Definition 40 (Adjacent qualitative states) Given a qualitative model described by the set of equations $\mathbf{E}$, two different qualitative states s_1 and s_2 are adjacent if one of their frontiers is common. If $s_1 = [v_1, v_2, ..., v_n]$ and $s_2 = [w_1, w_2, ..., w_n]$, s_1 and s_2 are adjacent iff $\exists! \, k \in [\![1, n]\!]$ such that $v_n \neq w_n$.

Computing complete qualitative behaviors makes it possible to represent it as a qualitative automaton.

Definition 41 (Qualitative automata) We consider the structure of a qualitative automaton $A_q = \langle Q, X, V, E, Init, I, F, J, L, A, T_Q \rangle$, where $Q, X, V, E,$ $Init, I, F, J, L$ are the elements defined in definition 8 and

- A is the system abstraction function mapping each mode m of Q to a partition of X where each set corresponds to a qualitative state of the system.
- T_Q is a function mapping each mode to the set of qualitative transitions computed in subsection 4.1.1.

This representation, introduced in [35] as abstract transition systems, proposes an alternative visualization of the behavior of a hybrid system, giving more information about the different qualitative trajectories in the modes. It can provide, for example, more knowledge about possible attractive states, intra-modal cycles, or even two qualitatively different trajectories that could not be separated using a classic hybrid automaton but which could imply various constraints for the system. This new layer of information completes our qualitative model with a more precise mapping of the qualitative states and gives more possibilities of anticipation, diagnosis, and piloting [34, 125].

4.3 . Problem of Dimensions

The presented process is simple to execute when $|X| = 2$ because polynomial solvers easily handle polynomial problems with two variables.

One of the components of X can trivially be expressed depending on the other for each polynomial function with at least one reference variable.

However, the presence of more dimensions raises problems that do not appear in two dimensions. The more $|X|$ increases, the more difficult it will become for the solver to return a usable solution. Especially symbolic solvers in high dimensions tend to find a unique solution that satisfies the constraints and will, therefore, avoid general solutions of the expected form. To ensure that the returned solution has the desired form, we must specify the anticipated solution format and the priority order of the variables for the resolution to the solver. The presence of solutions of dimension less than $|X| - 1$ will create frontiers that may be circumvented without being crossed, which is a major problem in computing qualitative transitions. Therefore, we added a filter to the solver that suppressed the solutions of low dimensions. Secondly, the choice of the component of X that should be considered as a reference to express the others is a question. Should X_0 always be regarded as the reference variable, or should there be a smarter decision criterion? Theoretically, the best solution would be to avoid the presence of fractions in the expressions of the solutions as much as possible. As fractions may cause singularity in the presence of a fraction pole in the state space, the best expression of a solution would be the one that minimizes the number of fractions. However, we made a concession due to the complexity of submitting such criteria to already implemented polynomial solvers. We just created a reference priority order favoring the lower factors of X to the upper. Finally, an inherent problem in this situation is that the used solver may not find any solution in high dimensions associated with high polynomial degrees. Even if polynomials of very high degrees are not commonly used in CPS, the situation may happen for specific systems and possibly in critical situations. This shows that there exists a possibility of generalizing this contribution even more, using elements from other works, such as qualitative tendencies.

5 - INTRODUCTION OF QUALITATIVE ZONES

The model obtained by the process presented in chapter 4 allows reasoning on the qualitative states and on the qualitative behaviors. This representation constitutes the heart of what is currently qualitative modeling. However, this representation of systems lacks knowledge about orders of magnitude and distance/time to an event that could allow the management of critical situations of the system. The exposed qualitative models represent the equivalent for the system state space of a topographic chart of the summits of a mountain chain with the altitude gradient between them. Yet, exploring such an important chain cannot be achievable without information about the gradient magnitude and the possible presence of cliffs. In a CPS, this is illustrated by the difficulty of making predictions or applying simulation on a model with no further information about the intended trajectories than the sign or the gradient. For an autonomous car, an acceleration of 1 m.s^{-2} and another directed in the same direction of 100 m.s^{-2} cannot be considered the same way in simulation or, worse, in real-time execution. Moreover, the same car will not behave the same way on an empty country road or a crowded freeway with other vehicles at less than three meters. Therefore, the two corresponding challenges are introducing elements from OMR [111, 113] in our qualitative models and adding thresholds surrounding and preventing the important events. As qualitative reasoning imposes constraints on the considered models, these upgrades to the representation must imply a process of refinement of the state space partition that matches the described ambitions. This means adding new borders to create more qualitative states, therefore adding new elements in the sets P_m. The inclusion of new polynomials in the P_m must be made with specific consideration to the interest and the significance of the added elements: adding more polynomials that correspond to no physical phenomenon does not have great interest. For example, creating qualitative states defined by the sign of the 5th order derivatives will rarely bring useful knowledge to the model. Yet, when all the equations describing the system and the most significant variable valuations have already been used to construct the previous model, the newly introduced equations have a very small chance of being of any significance to describe the system's behavior by themselves. Therefore, an option is to define the new border equations to link them to the previously defined ones to have a reasoning value associated with them. Modifying the elements p_i of P_m makes it possible to create two thresholds of borders surrounding the frontier defined by $p_i = 0$. These thresholds will then determine a neighborhood around the event frontier and describe a form of proximity before the possible occurrence of the related event.

The consideration of these neighborhoods, or proximity areas, will introduce information about the distance to the potential future events of the system behavior without clearly implying numerical evaluation. They can also have the role of precaution frontier, defining either a particularly far or critically close proximity to the event.

Now, let us see how to define such neighborhood limit (that we also call secondary frontiers, by opposition to the primary/main ones, which are supported by the elements of P_m). We mainly evoked and studied three main ideas. The first was to translate the main equations of P_m from a chosen distance d to obtain the neighborhood limit. The advantage of this idea is its simplicity of execution: each border from any dimension 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 may not have any physical meaning, especially in the case of borders defined by nullclines (here, the zeros of the derivatives of the system's variables and equations). This is, however, the best solution for the events defined by $X_i = k$, with X_i a component of X and k_i a constant in $\mathbb{K}$ defining a value for X_i: the translation direction is then naturally following the axis defined by X_i. The obtained hyperplane defined by the equation $X_i = k \pm d$ is parallel to the main border and represents a proximity area around it at a constant distance d. This solution is not possible in the case of nullclines supported by $\dot{X}_j = 0$.

Another idea was to compute surfaces completely parallel to the border: this would have assured a constant distance for every direction and would have been visually understandable. However, the computation of such surfaces is much trickier than expected: 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 time t by $x = f(t)$ and $y = g(t)$ with f and g two functions, the parallel curves to the parametric curve (x, y) are the parametric curves of equations (x', y') with

$$x' = f(t) \pm \frac{c\dot{g}(t)}{\sqrt{\dot{f}(t)^2 + \dot{g}(t)^2}} \qquad y' = g(t) \pm \frac{c\dot{f}(t)}{\sqrt{\dot{f}(t)^2 + \dot{g}(t)^2}}$$

c being a constant. This choice would make little sense as simplification and limited computation are the main objectives of qualitative modeling. Moreover, such a limit would not have any physical meaning. Consequently, this idea of parallel surfaces does not fit the ambitions of qualitative reasoning to compute a neighborhood limit for polynomial frontiers. Finally, the best of the raised propositions was to use isoclines (i.e., surfaces defined with a function f by $\dot{f} = c \neq 0$). For each element p of P_m, we must define a value $c_p \in \mathbb{K}$, then use the already defined 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: a very low proximity value will be associated with a critical proximity to the event and a very little time interval remaining before its occurrence. On the contrary, a very high value defining the neighborhood will inform that any state out of this area will be unstable and maybe untrustworthy as the associated variable or quantity will vary very fast.

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 defined depending on these latter, and that we called **qualitative zones**.

As the qualitative zones correspond to a subset of the state space defined by the proximity to a frontier, it is also possible to expand this notion to represent the neighborhood of a point. When representing the proximity of a chosen point x^p valuation of X, the proximity frontier of distance d is represented by a n-dimensional sphere of radius d of equation $\sum_{i=1}^{n}(X_i - x_i^p)^2 - d^2 = 0$. The corresponding qualitative zone is, therefore, the associated ball of the same dimension and of equation $\sum_{i=1}^{n}(X_i - x_i^p)^2 - d^2 < 0$.

Definition 42 (Qualitative Zone) A qualitative zone is a set of abstracted values in a neighborhood of a qualitative frontier or of a point. A qualitative zone is defined by a set of 2-tuple p_i, d_i with $p_i \in P_m$ corresponding to a qualitative frontier or the coordinates of a point and $d_i \in \mathbb{K}$ to the chosen neighborhood distance.

It is to note that some subsets of $\mathbf{X}$ may be included in many qualitative zones at a time: the creation of a function able to compute the intersection between qualitative zones is helpful to keep as much knowledge as possible about the proximity to the different events.

Suppose the qualitative states give abstracted knowledge about the distance from the current numerical state to each frontier. In that case, the qualitative zones offer an abstraction of the distance separating it from the nearest borders. This allows us to define a notion of distance in the qualitative models that may be used to evaluate the system's stability, the likelihood of a forthcoming event, or even the criticality of a situation. It gives a new tool to anticipate and reason about a system state, its risks, and possible futures.

Definition 43 (Qualitative Position) We call **Qualitative Position** of a system the 3-tuple $(mode, qualitative\_state, qualitative\_zone)$.

This complementary information creates a complete qualitative map of the system's state space designed to locate a numerical state and reason about its successors.

Finally, considering qualitative zones is a good criterion for separating two qualitative behaviors that could not have been distinguished by reasoning only on qualitative states. For example, in the Brusselator system defined in Equation 2.1 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, this knowledge is insufficient to deduce whether the trajectory will be convergent: the two qualitatively different behaviors (convergence and cycling around the convergence point) follow exactly the same trajectory among the qualitative states described earlier.

The consideration of a qualitative zone around the nullclines $\dot{x} = 0$ and $\dot{y} = 0$ and around the stable point allows us to determine, using the directional differentiation of the Lie derivative, if the system will converge towards $\dot{x} = \dot{y} = 0$, or stay in a critical cycle around it. With a symbolic expression of the distance d, the study of the surface defined by $||a - X|| = d$ or by $|\dot{x}| < d \wedge |\dot{y}| < d$ will show if the convergence is possible or not. If for any value of d, the inward transition is possible, and if from a threshold $d_s > 0$, the outward transition is impossible for $d < ds$, the system will be considered as convergent. On the contrary, if, for any qualitative state, there exists a distance d such that the inward transition is impossible, the system cannot converge and its trajectory will follow a limit cycle.

However, just like qualitative states, qualitative zones require the user to instantiate the parameters, such as the chosen proximity d, to be computed. As the automation of the choice of the number and the size of the qualitative zones has not been achieved, we did not integrate the creation of the qualitative zones in the abstraction process: we instead created a functionality to add qualitative zones on a qualitative model around the desired qualitative frontiers with a given size. Functions to test the intersection between a qualitative state and a qualitative zone and to compute the trajectory direction on the border of a qualitative zone are also available.

6 - IMPLEMENTATION AND EXPERIMENTATION

In this section, we develop a tool's structure and operation aiming at automating the algorithms presented in chapter 4 and chapter 5. This tool is implemented in *Python* to benefit from the various libraries available with the programming language. Firstly, we show its structure and its input and output elements in section 6.1. In section 6.2, we develop the implementation details of each tool block to present the choices and important elements of our implementation. In section 6.3, we show the results of our experimentation and develop the current limits and improvement directions currently identified.

6.1 . General Structure

Building a qualitative model requires passing a hybrid system as input to the program. It takes a structure *sys* passed as a list of elements that include the following items:

- Q a list containing all the discrete variables of the system and whose composition defines the operating mode.
- $modes$ the set of all possible valuations of Q (i.e. the set of all operating modes)
- q_0 the initial mode of the system
- var the list of all continuous variables of sys
- var_0 an initial valuation of var
- inv a dictionary structure associating to each mode q of $modes$ a list of invariant conditions that must always be satisfied. These constraints are represented using equations that must be compared to zero.
- tr a list of $4-tuples$ representing the discrete transitions of the system with for each transition, the initial mode, the guard condition, the target mode, and a mapping (represented using a dictionary) that associates to each element of var a reset constraint.
- F a dictionary associating each mode m to a mapping relating each variable of var to its dynamic function in mode m. The dynamic functions are supposed to be *ODE*s containing only polynomial or rational terms.
- par the set of symbolic parameters of the system.
- two structures containing the names of the variables and the system.

- Optional parameters, among which is a set of equations defining landmarks that correspond to specific use constraints or thresholds of this system instance and a set of specific objectives.

In order to perform the intended result, the program calls for the libraries *math*, *numpy*, *copy*, *sympy*, *Z3*.

6.2 . Functional Decomposition

6.2.1 . Model creation

From the input elements presented in section 6.1, the tool creates a first model that translates all the variables, parameters, equations, and constraints to *sympy* objects. The program calls the constructor of the *system* class, which contains *property* functions that allow us to access and modify all the presented characteristics that define a hybrid system.

If the constructor detects the presence of the optional parameter, the system is said to be instantiated as specific constraints, objectives, and requirements related to the targeted use case are provided. Otherwise, the system is not characterized.

Depending on the presence of these parameters, the obtained object will be an instance of one of two different classes whose difference is based on the quantity of information included in the definition of the system.

The first class, named *System*, considers a generic object that includes no information about its use case nor about its exact purpose or domain-specific constraints. The second one inherits the first and is called *Instantiated_System*. By the principle of heritage, all the knowledge accessible from a *System* object is also available in an *Instantiated_System* one, but the contrary is false. We chose to use object-oriented programming to impose a standard structure for the treated systems.

Once the first model of the system is created according to the requirements of one of these two structures, the analysis and discretization algorithm begins.

6.2.2 . State space abstraction

The obtained object is therefore passed as an argument of the function *qualitative_analysis*. This function first checks the nature of the entry class to determine if the system

- Is instantiated or not
- Has more or less than three continuous variables

The first criterion will decide if the program will look for landmark limits to be used in the discretization process, as the instantiated systems offer more varied knowledge than non-instantiated ones.

On the other side, testing the number of continuous variables of the system is necessary to determine the complexity of the study. As explained in section 4.3, the presence of three variables or more generates a supplementary obstacle in the choice of the reference variable to be expressed with respect to the others. Therefore, this condition will allow us to avoid useless computations in the case of a state space with two dimensions.

Once these elements are known by the algorithm, it will perform a discretization of the state space of the system for each mode $m \in \mathbf{Q}$.

This discretization is performed using the elements and properties presented in chapter 4. All the equations defining the system (including the landmarks in the case of an instantiated system) are covered in order and derived according to the process defined by Tiwari [35] to express the qualitative states. For each mode m, the discretization of the state space is done by placing all the polynomial equations that define the system in the set P_m and using the methods of symbolic solving provided by *Sympy* to resolve the equalities $p_i = 0$. The results of these equations define the parametric equations of the *nullclines* separating the qualitative states of the state space. *Sympy* also comes with tools to differentiate 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 of the set.

The results of the resolution of the borders are given as n-tuples with $n = |X|$, in the form $(f_0(X_i), ..., X_i, ..., f_{n-1}(X_i))$, that express the equations corresponding to the borders.

We then have a mapping associating each mode to a state space discretization based on the equations defining the dynamics, invariant, guard sets, landmarks, and the successive derivatives of all these equations.

In order to ensure the termination of the algorithm, one must add a stropping criterion in the abstraction function, given that the stopping criteria developed in [35] are not sufficient to guarantee that the program will terminate. Moreover, an over-discretization may cause the obtained model to be too complex for no tangible improvement. The naive criterion we added in this function to guarantee an end to the execution is a maximum number of derivations for each of the initial system's equations. This maximum can be either global or expressed as a set of values associated with each equation, and it will stop the Tiwari discretization process when a single function has been derived too many times. A further progress would be to consider the system's nature and application in order to deduce the most adapted granularity in the state space abstraction to automatically guide and stop the abstraction process when the optimal abstraction level is reached.

We also added a general constraint in the algorithm on the nature of the solution. If the solution satisfying $p_i = 0$ cannot be expressed as a fully differentiable function from the system variables X and requires discontinuity or piece-wise expression, we stop the derivation process of this polynomial at this iteration.

The result of the state space abstraction process is returned under the form of a mapping that associates each mode to a dictionary where the keys are strings referring to the nature of the initial equations (flow, guard, invariant, landmark, derivatives) and are associated with lists of 2-tuples containing both the polynomial/rational functions and the equations of the associated nullcline.

As *sympy* is perfectly able to deal with symbolic values and to solve equalities with both symbolic variables and parameters (which are not supposed to be valued nor solved) and, therefore, to express the solutions of the variables symbolically depending on the fixed parameters, our program can perfectly deal with partially designed systems (i.e., with parameters only expressed symbolically and without fixed numerical value).

If a state space refinement is needed, a new system based on the first one can be created using the *Instantiated_System* class. By calling the constructor of this class on the previous system and by adding the new equation constraints in the adapted list, we define a new iteration of the same system with constraints that will be taken into account in the state space abstraction, which will be a refinement of the first discretization.

6.2.3 . Numerical state abstraction

Once the state space abstraction mapping is chosen and available for use, the next component of our tool is the state abstraction function.

The program must be able to translate any numerical knowledge of the system to a qualitative state. Given a numerical value x of X and a current mode $m \in \mathbf{Q}$, for each $p \in P_m$, we compute the sign of $p(x)$ to determine on which side of each of the previously computed nullclines the current state is.

This is achieved by the function *abstraction* that takes as arguments the numerical value x to abstract, a valuation of the symbolic parameters of the system (necessary to allow comparisons with numerical values as *sympy* cannot handles inequalities with several unknown), the abstraction mapping obtained by the function *qualitative_analysis* detailed in the previous paragraph, the studied CPS and a negligibility threshold $\epsilon > 0$ under which a polynomial value will be considered equal to zero. Then, the sign of each $p \in P_m$ will be computed and pushed in a list in the same order as they are presented in the input mapping to have an easy correspondence between the signs and the associated equations.

Then, *abstraction* returns the qualitative state of the system corresponding to the numerical value and to the mapping obtained in *qualitative_analysis*. This qualitative state is represented by an array of -1, 0, and 1, corresponding respectively to $-$, 0, and $+$ but making operations easier to compute.

In our example of the Brusselator, 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 $[+, +, -, +, -]$.

This function is mainly helpful to abstract the initial state of the system given in parameter: given an initial numerical value x_0 and an initial operating mode m_0, computing the qualitative behavior of the system requires to convert this numerical position to a qualitative state from which the qualitative simulation will run.

6.2.4 . Qualitative state propagation

From here, the stake is to compute how the considered system will behave given all the known dynamics and constraint equations.

The first step is to compute the set of all qualitative states that share a common border with the initial state: they will be put in a list of *possible_successors*. Considering that the dynamics of the system are continuous inside a mode, the intermediate value theorem tells us that while no guard condition is violated, the only possible direct successors of a qualitative state are the qualitative states that share at least one border with it.

Therefore, we created a function named *neighbor_states* that takes as arguments the current qualitative state s_0 and the current mode m_0, the set of nullclines discretizing the state space in the mode m_0, a valuation of the system's parameters and the system itself. As two neighbor states are defined by the same constraints except the one corresponding to the shared border, the function furnishes the list of all the virtual neighbors of s_0 by taking the digit list that defines s_0 (composed of -1, 0, 1) and by returning the list of all the states defined by a list whose digits are equal to s_0 except one, which is modified taking into account the intermediate value theorem. This means that a 1 or a -1 can change to 0 and that a 0 can be converted either to a 1 or a -1, while a conversion from a -1 to a 1 would violate the continuity of the dynamics. In the case of a state $s_0 = [1, 1, -1, 0, -1]$, 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 $[1, 1, -1, 0, 0]$.

This list corresponds to a list of virtual successors, computed on the fact that states defined by these lists would be neighbors of s_0. However, nothing guarantees that these potential successors actually exist.

Consequently, we implemented a function testing the existence of these virtual successors. As the digit list of a virtual state vs expresses its position to the frontiers associated with the mode m_0, testing the existence of vs requires solving the constraints conjunction expressed by its digit list.

We then created a function *does_exist_state* taking as arguments the expression of vs, the nullclines equations defining the state space abstraction of m_0, and a valuation of the parameters of the system. As most of the constraints to solve are inequalities, we find here the limits of *sympy* as it does not allow solving inequalities containing more than one unknown variable. Consequently, we made the choice of using a *SMT* solver available in Python to solve this constraint conjunction. We chose *Z3* for its simplicity and permissivity. However, using such tools imposes to translate all the inequalities and equalities defining the constraints from *sympy* expressions to *Z3* ones, as the two of them are not compatible.

We achieve this translation with a function *sympy_to_z3* taking as arguments the *sympy* expression to translate as well as the list of variables involved in this expression. Once translated, it is possible to use the SMT solving functionality of *Z3* to determine whether the conjunction of constraints associated with the virtual state vs admits a solution. The *does_exist_state* calls another function called *solve_intersection* solving the inequalities defining vs by transforming the $-1, 0, 1$ digits to $<, =, >$ operators. This function uses *Z3* and returns *True* if there exists $x \in \mathbb{K}^n$ that satisfies all the constraints defining vs and *False* otherwise. If the problem is satisfiable, vs corresponds to a concrete set of values of the state space of the system and therefore exists. In that case, vs is added to a list of *possible_successors*. Otherwise, vs does not exist and is therefore not further considered.

As we know for each of the considered neighbors, which digit has been changed from s_0, and that the order of the digits corresponds to the order of the abstraction equations, the algorithm can know the nature of the common border that will be crossed. If this frontier happens to correspond to an invariant violation, we keep the new state in a special category: we will still study the feasibility of the transition but keep it as an invalid one.

Once all the neighbors of a qualitative state s_0 have been isolated, the program must determine which are successors and which are predecessors. As explained, it achieves this by computing the Lie derivative of the equation corresponding to the border. To this extent, it calls a function called *is_allowed_transition* taking as arguments the state s_0, the target state s_t, the set of abstraction equations, the parameter valuation and the mode in which the transition is occurring.

It executes it with a scalar product between $\left[\frac{\partial p}{\partial X_i}\right]_{X_i \in X}$ and $\left[\frac{\partial X_i}{\partial t}\right]_{X_i \in X}$, with $p = 0$ defining the border. Ideally, this should be evaluated on a numerical value x of X located in the current state to determine this scalar's sign. However, regarding the difficulty of finding a satisfying concretization function expression (which is still one of the biggest challenges of qualitative reasoning), there is not yet a possibility of getting numerical values corresponding to a qualitative state.

In order to bypass this obstacle, our function creates 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 digit whose value depends on the sign variation implies by the considered transition. This new element is linked to the equation obtained with the Lie derivative formula. Then, we use the *Z3* SMT solver once more to determine if this virtual qualitative state exists. If the associated problem is satisfiable, then the Lie derivative on the border allows the transition, and the state s_1 considered as a possible s_0 successor is an actual successor of s_0.

Finally, we wrote a function *select_allowed_transitions* that, given a mode m and a current state s_0, automatizes the developed process to compute every successor of s_0 and returns them in a list.

6.2.5 . Discrete transition management

Once all the successor states of a given qualitative state have been found, the problem of the states violating the guard or the invariant conditions still exists.

As the nature of each qualitative transition can be easily computed using the modified digit between the initial and target qualitative states, we wrote a function *transition* that detects which qualitative transition from an input list violates a guard condition and, if needed, which guard condition. With this knowledge, and using our hypothesis that the modal transitions are deterministic, the program refers to the system definition and exposes the transition that happens under this condition. Therefore, if the qualitative state corresponding to the border of the guard set is kept, the states that clearly violate a guard equation are forgotten and replaced by a state in the target mode corresponding to the event triggered by the constraint.

If the associated reset function is the identity function $Id : X \mapsto X$, the *transition function* calls new instances of a *Z3* solver to generate digit by digit the qualitative states of target mode m_1 and its associated abstraction equations. As the state space discretizations of m_0 and m_1 are likely different, transposing without computation a qualitative state from m_0 to m_1 is impossible. Therefore, our function generates using SMT solving all the qualitative states s_i form m_1 that overlap with m_0. Therefore, the qualitative states defined by (m_1, s_i) will take the place of the initial guard condition violating state as successors of s_0.

For the transitions that do not have the identity function as their reset function, we call another method called *correspondance_transition* then computes the image of the initial qualitative state s_0 by the reset function on the state space abstraction of the target mode m_1. Just as in previous functions, it uses the SMT solver functionality of *Z3* to compose digit by the digit the qualitative states of m_1 that overlap with $(res(s_0))$.

Consequently, our tool is now capable of dealing with discrete transitions and reset functions and can represent the qualitative evolution of hybrid systems.

6.2.6 . Qualitative behavior computation

As the previous blocks of our tool are able to compute all the successors of a given qualitative state in a given mode and to manage the mode switch, the final step is to compute the complete qualitative behavior of a system. As we aimed, we created a new class of objects named *Qualitative_automaton* containing all the equations defining the initial system, a set of modes, and a set of qualitative states, both accessible by getters and setters and initialized with the initial qualitative state and mode of the system and a dictionary structure initialized empty.

This dictionary aims to represent the qualitative trajectories of the system. After its initialization, all the modes of the system are passed as keys, and the element mapped to m_0 is another dictionary structure where each explored qualitative state will be mapped to two arrays representing its predecessors and its successors, respectively. The *predecessor* array of the initial qualitative state is initialized with the element *start*.

The successors of the initial state s_0 are then computed and put in a queue called *Frontier*. For each of them, it is added in the *successors* structure of s_0, in the set of qualitative states of m_0 (and therefore as a key of $system[m_0]$) and s_0 is put in their predecessor's list. A *visited* list is also initialized containing the initial state (m_0, s_0).

While the frontier is not empty, the first element s_1 is popped out of the structure and becomes the current node if it is not already in *visited*. Just as for the initial state, all its successors are computed and added to its successor's list, while s_1 is added as a predecessor for all of them. If s_1 happens to violate invariant conditions, it is not added at all in the behavior tree, but a state named *Invariant Violation* is added as a successor of $s1$. In the case of qualitative state implying modal transitions, the function *correspondance_transition* defined earlier is called to make the correspondence between the initial and the target mode. In this situation, the target state is designed as the 2-tuple (m_1, s_1) in the *successors* list of s_0 in m_0 to highlight the discrete transition, and s_0 is written (m_0, s_0) in the *predecessors* list of s_1 in m_1 for the same reason.

Once the frontier is empty, it means that the complete connex structure of the behavior of the system leaving from this given initial state has been reached. The algorithm then gives this graph as output.

Qualitative states may exist that have not been explored, but the exploration structure implies that these states are not reachable from the chosen initial point.

6.2.7 . Inclusion of the qualitative zones

Once the complete qualitative automaton containing the path representing the qualitative behavior of the system from an initial value x_0 in an initial mode m_0 is computed, we developed a set of functions and a new object class to add qualitative zones in a common structure allowing a more exhaustive study of the system behavior.

The first of the new set of functions is called *create_zone_quali* and takes as parameters a distance d and either a point or a frontier among the borders separating the qualitative states of a mode. It melts them into a 2-tuple that will then represent the qualitative zone centered on either the point or the frontier and defined by a distance d around this structure.

The second function is *generate_equation_zone* as it takes as a parameter the previous tuple and gives as output the equation corresponding to the borders supporting the qualitative zone. If it is supported by a state-frontier, the zones equations will be deduced from the initial nullcline equation $p_e = 0$ by computing $p_e = \pm d$. Otherwise, if the researched zone is centered on the point of coordinates $X = x$, then the frontier is given by a n-dimensional sphere supported by the equation $(X_0-x_0)^2+(X_1-x_1)^2+...+(X_n-x_n)^2 = d^2$ if we suppose that the used coordinate system if Cartesian. The obtained set of equations defines the subspace of the state space that is included in the created qualitative zone.

The next functions, respectively called *is_in_zone*, *coexist*, *overlap* and finally *can_enter_zones* aim at improving the understanding of the behavior of the system by positioning the qualitative zones relatively to numerical values, to qualitative states, to other zones and finally to express if a qualitative behavior can enter in a qualitative zone using the Lie derivative equations as developed in subsection 6.2.4.

Finally, the newly created zones are placed with the previously computed qualitative automaton in a new structure called *qualitative_automaton_z* that inherits directly from *qualitative_automaton* and authorizes to add qualitative zones, which will be studied to improve the quality of the system representation. The *coexist* function computes all the qualitative states that share some subsets with a qualitative zone, while *can_enter_zones* will compute if entering the zone is possible from a given state that coexists with it.

Using these functions, it is possible to be even more precise in the qualitative behavior and to prove some properties, such as the convergence of the trajectories of the system.

6.3 . Experimentation and Limits

In this section, we will present the results of some experiments on different categories of systems at different abstraction levels.

6.3.1 . Hybrid systems

Applied to the hybrid Brusselator presented in example 11 on page 71, the algorithm gives the qualitative automaton shown in Figure 6.1.

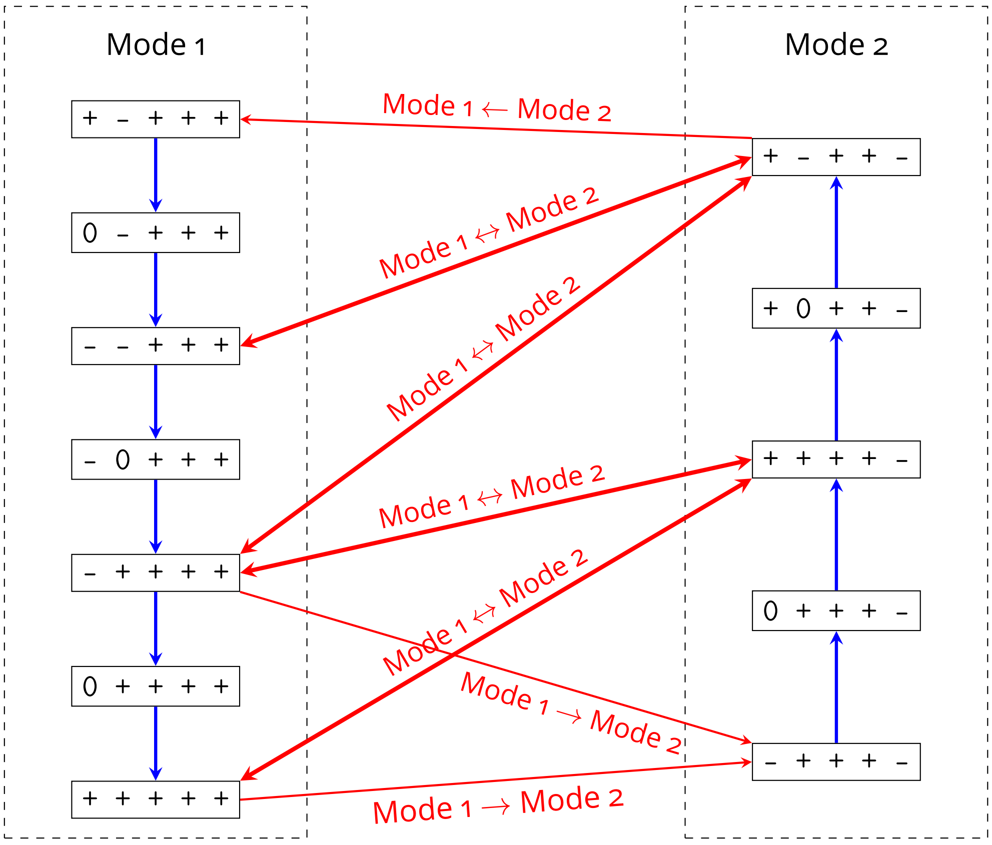

Figure 6.1: Complete qualitative automaton of the hybrid Brusselator system

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. Qualitative transitions are all unidirectional, so this distinction does not apply to them. 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_m. With a stopping criterion for the discretization of the state space of at most one derivation for any polynomial of the model, P_m is equal to $\{1-(b_m+1)x+a_mx^2y, b_mx-a_mx^2y, x, y, x-y\}$. The first two items correspond to the flow equations of mode 1 and mode 2, the next two items to the invariant constraints, and the last item to the guard condition. In each qualitative state, the tuple of elements from $\{-,0,+\}$ characterizes the sign of the corresponding elements from P_m that define the state as explained in definition 31. For example, the qualitative state represented by the digits $(+,-,+,+,+)$ is characterized by the inequalities $1-(b_m+1)x+a_mx^2y>0, b_mx-a_mx^2y<0, x>0, y>0, x-y>0$. Many qualitative cycles coexist, all of them transiting by the two modes. There is no behavior staying in only one of them. The existence of many different cycles shows the interest of the state space discretization, as it separates trajectories that would have been considered identical in a classic hybrid automaton. We now have the complete qualitative automaton of the CPS. Both the discrete transitions and intra-modal behavior are included, and the trajectories are more visible than in traditional hybrid automata.

6.3.2 . Qualitative zones

Once the qualitative automaton is obtained, the following step is to compute the qualitative zones around the qualitative frontiers and their different interactions with qualitative states. The distances d_i between every qualitative border and its associated secondary frontier must be entered as an argument: we still need to automatize the choice of the value characterizing the optimal distance. Once we have the equations of the different hyperplanes and isoclines, we aim to determine how they fit into the qualitative model. This involves a new call of the constraint solver: the intersection between qualitative states and zones can be computed by testing the satisfiability of the conjunction of the constraints associated with the intersection of a state with a zone. If the problem has a solution, then the considered qualitative zone exists in the qualitative state. Knowing which zone exists in each qualitative state and computing the transition direction once more creates a qualitative map that gives more 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 Equation 6.1, where b is a constant parameter.

$$\begin{cases} \dot{x} = 10(y + x - \dfrac{x^3}{3}) \\ \dot{y} = b - x - \dfrac{3y}{4} \end{cases} \tag{6.1}$$

This system's few equations and borders make it an excellent example, as the qualitative map is still understandable and not overloaded. In Figure 6.2, we drew the main borders with plain lines and the associated secondary bor-

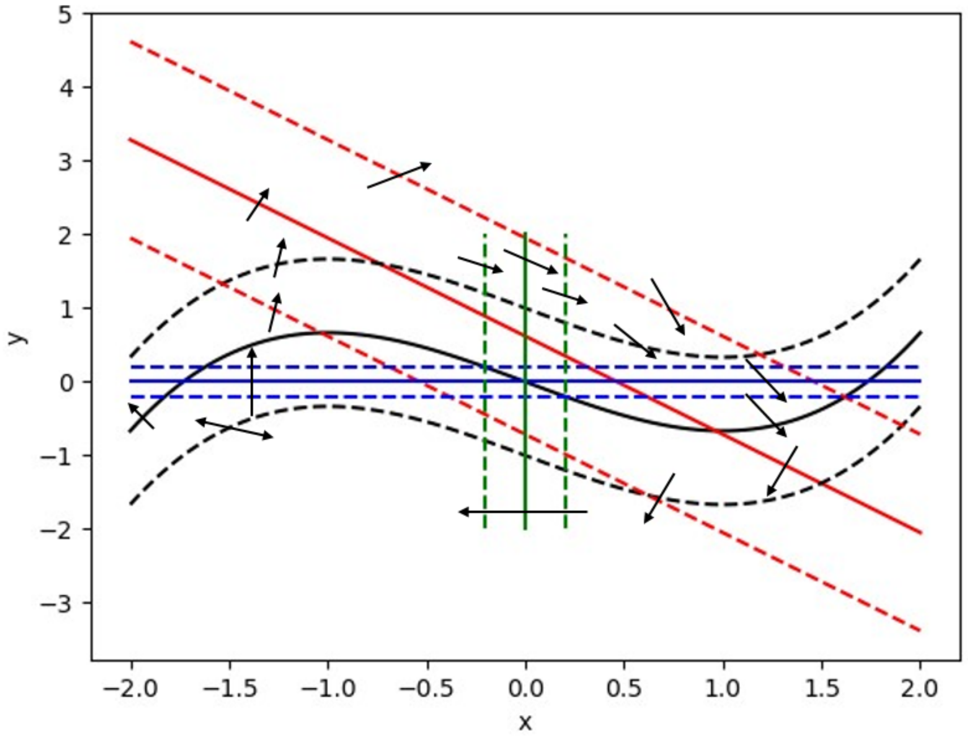

Figure 6.2: Qualitative map of a Van der Pol oscillator system

ders with dotted lines. This representation corresponds to an instance of the system where b is arbitrarily set to 0.465. The color code associates each border with its secondary 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 allowed according to the computation of the Lie derivatives. Double arrows highlight that both transition directions are possible at a point of the border.

7 - APPLICATIONS

This chapter presents different applications of qualitative modeling and reasoning about CPSs that would benefit from the simplification and generalization allowed by the state space and behavior abstraction. Section 7.1 evokes how qualitative models can be used for operations such as reachability tests or behavioral proof. Section 7.2 details the possibility of creating a qualitative pilot to supervise and optimize numerical simulations using qualitative positions. Section 7.3 generalizes this idea to system monitoring. Finally, section 7.4 presents some propositions about the possibility of using qualitative reasoning to solve the DSE problem for a system with undefined parameters.

7.1 . Reachability, Verification, Proof, Diagnosis, Test

7.1.1 . Reachability

As we are now able to generate an exhaustive tree of the qualitative behavior of the system, it is possible to use this vision of all theoretically possible behaviors to study the reachability of a set of states by the modeled system.

Given initial conditions and a set of qualitative states represented by the same equations the model uses, one just needs to browse through all the keys of the qualitative automaton structure to know if the given states exist. As qualitative states are added in the structure once they are reached during the qualitative simulation, the presence of the state s_i among the list of keys of $auto[m_j]$ means that the qualitative state defined by the couple (m_j, s_i) is reachable by the system if initialized at the given initial condition c_i. Therefore, a simple algorithm as written in Algorithm 1 is sufficient to prove or contest the safety of the system according to some specified constraints.

It is to be noted that if the equations supporting the tested constraints are not equations that intrinsically define the system, one must use an instantiated system and add this equation to the case-specific constraints in order to use them in the discretization process.

The *rstates* structure is presented as a dictionary associating to each operating mode a list of qualitative states whose reachability must be tested using the form of the digit vector presented in chapter 4.

Using the method *get_states* implemented inside the *qualitative_automaton* class, we can get for any mode the list of qualitative states visited in this mode. For each mode m, the algorithm just needs to search through the list returned by *automaton.get_states(m)* and to return a structure similar to the input *rstates* replacing each state by a boolean indicating its reachability or not.

As the two unconditional loops are designed to browse through the input

```
Data : $v_{init}, m_{init}, rstates, system, parameters, quali\_automaton$
Result : Boolean values expressing the reachability of each of the
         input qualitative states
$answer = rstates$
for $m \in keys(rstates)$ do
    $reachable\_states = quali\_automaton.get\_states(m)$
    for $i \in [\![0, length(rstates[m])]\!]$ do
        if $rstates[m][i] \in reachable\_states$ then
            $answer[m][i] = True$
        else
            $answer[m][i] = False$
        end
    end
end
return answer
```

Algorithm 1 : Reachablity search.

structure *rstates*, and as an inclusion test is performed for each of its elements to the set of reachable states of the qualitative automaton, this algorithm's complexity is $O(rh)$ with $r = |rstate|$ and $h = |qualitative\_automaton|$.

7.1.2 . Verification and conformance checking

The possession of the qualitative automaton also allows us to do verification and test the conformance of a system to the predicted behavior. If one disposes of a measured trajectory of a system, even obtained by simulation or by measuring the execution of a concrete system, one can compare the corresponding time series to the qualitative behavior computed by our algorithm and use abstraction functions and transition tests to validate or dismiss the system from the measured data.

Given a time series of measurements made for each continuous variable of X and the knowledge at each moment of the time series of the associated operating mode (that we suppose known by the software component of the CPS), we can compare the accuracy of the time trace to the supposed qualitative behavior as presented in Algorithm 2.

If the measured trajectory matches a qualitative trace among the qualitative behavior tree, it means that the measured behavior is compatible with the physical theory and that the system may be valid. Otherwise, it means that something happened in the system's execution, preventing it from behaving as supposed, and therefore, the system is not reliable.

To verify the concordance between a trajectory and the qualitative behavior, the algorithm abstracts each point of the trajectory after the other and checks if it corresponds to the same qualitative state as the previous one. If

```
Data : $trajectory, system, parameters, Bt$
Result : A boolean value indicating if the input trajectory is possible
         given the definition of a system
$auto = hybrid\_automaton(system, trajectory[0][state],$
 $parameters, trajectory[0][mode])$
$discr = abstraction_{space}(system)$
$ab = abstraction_{val}(trajectory[0][mode], trajectory[0][state],$
 $parameters, system, Bt)$
$last\_state = trajectory[0]$
for $point \in trajectory[1 : end]$ do
    if $point[mode] = last\_state[mode]$ then
        $qs = abstraction_{val}(point[mode], point[state],$
         $parameters, system, Bt)$
        if $qs \neq last\_state[state]$ then
            $succ = auto.get\_successors(point[mode],$
             $last\_state[state])$
            if $qs \notin succ$ then
                return False
            end
        end
    else
        $qs = abstraction_{val}(point[mode], point[state],$
         $parameters, system, Bt)$
        $succ = auto.get\_successors(last\_state[mode],$
         $last\_state[state])$
        if $(point[mode], qs) \notin succ$ then
            return False
        end
    end
end
return True
```

Algorithm 2 : Conformance testing

it does, nothing needs to be done, and the program will move to the next point. Otherwise, it refers to the qualitative automaton to know if there exists a qualitative transition from the qualitative state corresponding to the previous point to the one associated with the current value. If the transition is allowed, the execution continues. If it is not, it means that a transition that is not allowed by the system's equations appeared in the time series. Therefore, the algorithm returns that this trace is not allowed by the automaton and that it is incorrect.

In the shown algorithm, the variable Bt corresponds to the thickness parameter of the qualitative borders: as there is very little probability that the time of all the measures corresponds exactly to the crossing of any qualitative border, it is very likely that considering the borders as regular lines (which they actually are) would lead to direct transitions from qualitative states of both sides of the frontier. However, as this violates the intermediate value theorem, the corresponding trajectories may be considered false, or the algorithm should look forward to two transitions instead of one to verify this situation, which would increase the time complexity of the program. To deal with this situation, we consider the borders as thick structures composed of the real frontiers and of an uncertainty neighborhood that would encompass any point in this area as being on the border. The thickness of this area is determined by the parameter Bt, the value of which must be chosen according to the frequency of the measures contained in the trajectory in order to avoid any possible unmeasured qualitative transition.

As this algorithm is a loop on each numerical position of the input trajectory, which computes the associated qualitative state for each of them, it has a complexity of $O(|trajectory| * complexity(abstraction_{val}))$. The complexity of the abstraction function directly depends on the number of qualitative borders that define the qualitative state space. Although this value cannot be expressed in the general case, it strongly depends on the number of continuous variables of the system and on its number of discrete transitions (itself depending on its number of discrete variables). More specifically, the number of elements in the set of qualitative borders can be approximately expressed as $kn + q$ with $n = |X|$, $q = |\mathbf{Q}|$ and k a positive constant. This directly results from the abstraction algorithm that mainly uses dynamics equations from $|X|$ and their derivatives, as well as invariant conditions and guard conditions. Therefore, if we note l the number of points in the input trajectory, the complexity of the conformance testing is $O(l(n + q))$ with $n = |X|$ and $q = |\mathbf{Q}|$.

7.1.3 . Proof of properties and fault diagnosis

As exposed in [26, 34], qualitative reasoning has an important added value in fault diagnosis. Using qualitative reasoning and simulation to create a qualitative model and map of the system's state space can not only be used to explore the set of all the reachable states in the propagation order but also to use back-propagation of the qualitative states in the behavioral trees in order to trace back the initial error or deviation that ultimately lead to a malfunction or a faulty behavior.

We mention this application of qualitative reasoning, further developed in a previous thesis in the CEA [26]. Using a qualitative model, the authors applied a methodology named *CEGAR* (for CounterExample Guided Abstraction Refinement), originally developed to check safety properties and based on the refinement of the state space abstraction using Counter examples defining faulty behavioral results. They also develop the concept of twin-plant diagnosis, using a non-deterministic automaton that represents the available knowledge about the system based on its observable events.

The interest of qualitative reasoning in diagnosis is the plasticity of the discretization of the state spaces (represented only by arrays of comparison operators or digits) that can be refined to add constraints and examples using polynomial or rational inequalities on the variable values.

Following the same fundamental philosophy, qualitative reasoning can play an important role in the proof or invalidation of specific properties of systems, such as convergence or periodicity of a system.

A simple look at the qualitative automaton of a system such as the standard brusselator highlights the cyclic nature of the system.

If it does not prove the exact periodic nature of the numerical trajectory, it shows that the behavior is qualitatively periodic in the sense that the same sequence of qualitative state is to happen in the same order, be the system convergent or not. Moreover, adding some specific constraints with abstraction refinement allows us to define qualitative states and qualitative zones whose reachability can be studied to prove more complex behavioral properties of the system.

For example, knowing the state space areas where the dynamic functions are contractive and those where they are expansive would be of great interest to increase the capabilities of numerical simulation. According to the definition, a function is a contraction if it is k-Lipschitz with $k < 1$. A function f is k-Lipschitz iff $\forall (x, y) \in D_f^2,\ |f(x) - f(y)| \leq k\,|x - y|$. It can be proved that f will be contractant if $|\dot{f}| < 1$. Therefore, by creating the qualitative zone defined by the distance $d = 1$ around the nullclines associated with the equations $\ddot{X}_i = 0$, the qualitative model includes the areas of contractive behavior of the system's dynamics. In this area, a simulation can be allowed to be less precise as the uncertainty area of the simulation will contract to the real behavior.

Moreover, qualitative zones can also be used to improve the reasoning toolbox of the qualitative model and discriminate between different behaviors that would have been qualitatively identical otherwise. The cases of the

Van-der-Pol and the Brusselator systems are particularly interesting from this perspective. Using only knowledge brought by operating modes, qualitative states, and transitions, one would not be able to know if an instance of the system is convergent or if it reaches a limit cycle, as the two possibilities are represented using a qualitative periodic behavior. As a convergence point implies the derivatives of the system's variable to all be equal to zero at the same time, a system will only converge to the intersections of its nullclines defined by the equations $\dot{X}_i = 0$. Consequently, in the case of a system like the ones cited above, the proof of the convergence or critical stability of the system can pass by the definition of a set of qualitative zones defined around the critical points $\dot{X} = 0$ defined by the inequality $\sum_{i=1}^{n} d(x_i, c_i)^2 < \epsilon$ with c_i the coordinates of the stable point, $d(.,.)$ a distance defined on $\mathbf{X}$ and $\epsilon > 0$.

Therefore, if for any $\epsilon > 0$, the computation of the Lie derivatives concludes that it is possible to enter the qualitative zones centered around the equilibrium point, it means that the critical point is a stable equilibrium and therefore that the system may converge to this point. Otherwise, if for ϵ sufficiently small, the Lie derivative does not allow the system to enter the qualitative zone defining the neighborhood of the critical point, it implies that this equilibrium is unstable and that the system shall not converge.

Moreover, if one can determine the limit value of ϵ from which entering the qualitative zone of distance ϵ from the critical point becomes impossible, one can deduce the numerical trajectory and the coarse shape of the limit cycle in which the behavior will be trapped.

These examples show that using qualitative zones to complete the previously defined qualitative models can allow us to discriminate behaviors that would have been qualitatively equivalent otherwise and, therefore, add reliability to the analysis authorized by our models.

Example of formal proof using qualitative reasoning

To illustrate the capability of a qualitative model to prove some system properties, let us consider a simple system composed of two cars advancing on a road following an axis x. The considered system is then represented by a set of two continuous variables $X = (x_1, x_2)$, with $Q = \varnothing$. Considering that at time $t = 0$, the first car is at $x_1 = 0$ and the second one is at $x_2 = d > 0$, the second car is ahead of the first one by an advance of d. The system's dynamics are defined using the differential equations:

$$\begin{cases} \dot{x}_1 = c_1 \\ \dot{x}_2 = c_2 \end{cases} \tag{7.1}$$

with $c_1 > c_2$.

Then, it appears evident that the second car will overtake the first one and stay ahead until the equations of the dynamics change.

By computing the qualitative automaton of this system using three qualitative borders defined by the nullclines $x_1 = 0, x_2 = 0$ and $x_1 - x_2 = 0$, we obtain the following automaton:

$$\begin{tikzcd}
\boxed{0\ +\ +} \arrow[d] & \boxed{+\ +\ -} \arrow[loop above] \\
\boxed{+\ +\ +} \arrow[r] & \boxed{+\ +\ 0} \arrow[u]
\end{tikzcd}$$

Figure 7.1: Qualitative automaton of the two cars with constant speeds

This result corresponds to the expectations as the initial state where the first car is at $x_1 = 0$ is repellent, while its two successors are transitive. The final state, on the contrary, is absorbent: the first car, once ahead of the second one, cannot return behind.

It is also possible to consider a more complex version of this system where the speeds of the vehicles are not constant but evolving linearly with constant accelerations.

This system is defined with four continuous variables, corresponding to a set $X = (x_1, x_2, \dot{x}_1, \dot{x}_2)$, with the initial conditions $(d > 0, 0, v_1 > 0, v_2 > v_1)$.

This new system would be defined by the equations:

$$\begin{cases} \ddot{x}_1 = c_1 \\ \ddot{x}_2 = c_2 \end{cases} \tag{7.2}$$

with $c_1 > c_2$.

This new system should exhibit an uncertain behavior where the first car may be temporally overtaken by the second one but could also stay ahead depending on the values of c_1 and c_2. It is, however, certain that after some time, x_1 will definitely be higher than x_2.

For this system, considering the nullclines $x_1 = 0,\ x_2 = 0, x_1 - x_2 = 0, \dot{x}_1 = 0, \dot{x}_2 = 0, \dot{x}_1 - \dot{x}_2 = 0$, we obtain the qualitative automaton represented in Figure 7.2.

Once again, the obtained results correspond to the expected behaviors and prove that, independently from the numerical behavior, the finality of such a system is that the first car will always finish ahead of the second one.

7.2 . Supervised Quantitative Simulation

The process of qualitative model creation can now be partially automatized to generate a hybrid automaton representing the system's behavior. If qualitative models are mainly used for theoretical and upstream tasks such

$$\begin{tikzcd}
\boxed{+\ 0\ +\ +\ +\ -} \arrow[d] & \\
\boxed{+\ +\ +\ +\ +\ -} \arrow[r] \arrow[d] & \boxed{+\ +\ +\ +\ +\ 0} \arrow[d] \\
\boxed{+\ +\ 0\ +\ +\ -} \arrow[d] & \boxed{+\ +\ +\ +\ +\ +} \arrow[loop right] \\
\boxed{+\ +\ -\ +\ +\ -} \arrow[d] & \boxed{+\ +\ 0\ +\ +\ +} \arrow[u] \\
\boxed{+\ +\ -\ +\ +\ 0} \arrow[r] & \boxed{+\ +\ -\ +\ +\ +} \arrow[u]
\end{tikzcd}$$

Figure 7.2: Qualitative automaton of the two cars with constant accelerations

as verification, proof, and diagnosis, our contribution of adding the concept of qualitative zones to the qualitative model gives a glimpse of possible concrete applications of qualitative reasoning on concrete CPS, even once already designed and generated.

The second application we present here is the use of the qualitative map (including the qualitative positions) to optimize the performances of numerical simulation. Guided by the knowledge encompassed in the map, we can guide the execution of a simulation to improve the quality-cost ratio.

More specifically, we intend to solve one major drawback of numerical simulation, which is the regular need for rollback to precisely localize an event that the simulation just passed without knowing exactly its occurrence time t_e. Qualitative maps will give improvement possibilities to prevent the occurrence of the qualitative events and, therefore, adapt the simulation time step to locate the event moment as precisely as possible without having to rollback.

Improving the execution of a numerical simulation can be achieved by upgrading different parameters. In our studies, we focused on the computation time and the quantization precision.

The reduction of computation time can be easily influenced by modifying the simulation time step, which can be done with our newly obtained model to improve the adaptation strategy and preserve the precision of the results.

7.2.1 . Adaptive time steps

Adapting the step of a numerical simulation to the situation is something classical. The idea has been expressed in many works such as [126, 127, 128] on various types of systems and dynamics, and is even applied to stochastic differential equations [129]. As explained in [130], adaptive step algorithms consist of defining a measure to control the integration time step at each iteration. This measure can be the estimated precision of the simulation process, the intensity of the derivative values at each iteration, the mean value, and the standard deviation of a potential stochastic term. It must be used to adapt the integration step to the situation.

Standard integration methods such as Euler or Runge-Kutta have been used as bases to develop adaptive steps processes.

Using an adaptive integration step for computing the value of the state variables over time can both reduce the number of required steps to reach the end of the simulation and also improve the precision in the critical areas where we want to locate the occurrence of qualitative events.

Where some programs make a first simulation to get an approximation of the result and make a second run with an adapted integration time step [131], we prefer a single-run method, which is suitable for resource management and 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 system's qualitative model to determine when events occur.

Considering that we mainly deal with *ODE*s and that the very nature of qualitative abstractions prevents us from comparing the computed system state to a reference value, the major option we have left is to adapt the simulation step to the value of the derivative with respect to time $\frac{dX}{dt} = \dot{X}$ (that contains knowledge about the variation direction and speed). Moreover, the values of $\dot{X}_i$ are among the knowledge that is intrinsically defined by the qualitative zones as developed in chapter 5. The frontier of a qualitative zone is indeed defined using the isocline equation $p_k = \pm c$ with $p_k \in P_m$ for a mode m and c a constant value.

Therefore, we made the choice of developing a new abstraction function α_z that abstracts for each mode the numerical state of the system on the maps of qualitative zones. If we note $\mathbf{Z}_m$ the set of all the chosen qualitative zones on the mode m, the abstraction function α_z is defined as

$$\begin{aligned} \alpha_z : \quad & (\mathbf{Q}, \mathbf{X}) \to \mathcal{P}(\mathbf{Z}_m) \\ & (m, X) \mapsto \{q_z\}_{X \in q_z} \end{aligned} \tag{7.3}$$

The output of α_z is the set of qualitative zones defined on the mode m of the system S in which the numeric value X stands.

As qualitative states (and therefore qualitative zones) are characterized by the nullclines/isoclines that define them, the knowledge of $\alpha_z(m, X)$ for any $(m, X) \in (\mathbf{Q}, \mathbf{X})$ gives information about the event that is about to happen, and about the magnitude of the derivative values $\dot{X}_i$ for every component of X. If X is in a zone corresponding to the proximity of an event (symbolized by the inequality $|p_k(X)| < \epsilon$ with $p_k \in P_m$), it can be considered that the qualitative border supported by p_k may be crossed and therefore the corresponding qualitative event may be triggered, which could imply a change in the dynamics or in the variation direction.

On the contrary, if X is in a zone defining a high value of at least one component of $\dot{X}$, the risk is that the current simulation step may be insufficient to encapsulate the strong variation of X in the steps to come.

Both cases require reducing the integration step h to adapt to the need for more precision in these precise areas in order to avoid rollback and loss of knowledge that would be irreversible.

7.2.2 . Time step and maximum variation

One of the major drawbacks of qualitative reasoning in the case of numerical simulation is that once knowledge has been lost, there is no reliable method to earn it back. One method to adapt the precision level with time is to switch models depending on the amount of estimated accuracy of the simulation, but it requires being able to measure the error and, therefore, to know the exact state of the system in some specific instants, which is not possible in many cases.

The usual way to adapt the integration step to the variation speed of the state variables is to use the largest integration step that mathematically guarantees a bounded error on the computed value. The main factor of error in numerical simulation comes from an integration step that is too large with respect to the variation speed of the system's values, represented by the derivative values $\frac{dX}{dt} = \dot{X}$.

To simplify the computations, we chose to reason with a discretized representation of the time scale represented using decimal numbers (i.e., the scale used to define our integration steps is defined as the set $\mathbb{D}$ of numbers that can be expressed as $\frac{a}{10^k}$ with $a \in \mathbb{Z}$ and $k \in \mathbb{N}$.

As exposed in the previous paragraph, we had a double objective: making sure to avoid rollback and optimizing the computational resources of the simulation.

In order to force the simulation to adapt its time step to the gradient values, we impose the value $\lceil \log_{10}(|h\dot{X}_i|) \rceil$ to be upper bounded to a reference value for each $X_i \in X$ with h the current integration step. As automating the choice of this value would require the model to analyze complex information such as the system's nature, its supposed application, or the units of the implied physical units, we require this value to be imposed by the user. They must be chosen according to the distance defining the different nullclines around the qualitative borders to have a reference time step coherent with the evolution of the system.

To prevent any qualitative transition from being missed, it must be made impossible to cross a qualitative border (and therefore change the qualitative state of the system) without stopping in the neighborhood of the associated frontier defined by the proximity qualitative zone. This criterion ensures that the system will not have an unplanned transition, which would require a rollback of the simulation to detect its exact spot. We made the choice of a $\log_{10}$ gradation to be consistent with the time scale discretization and to stay close to human reasoning (which is a key element of qualitative reasoning as exposed in [63]).

Some experiments were achieved using bases such as 2 or 16 instead of 10, and it appears that smaller bases improve the precision while bigger ones reduce the number of simulation steps. Base 10 makes the program structure more intuitive and understandable and fits qualitative reasoning philosophy while fitting a compromise between the two objectives. The choice between these possibilities will depend on the objective of the model and the preference between efficiency and simplicity. As every model does not have the same requirements and variation speed and amplitude, some systems will naturally favor one choice over the others.

Consequently, when one of the components of $|\dot{X}|$ is too high, the qualitative pilot will automatically slow down the simulation by reducing by a factor 10 the integration step.

On the contrary, when the pilot finds that the term $\lceil \log_{10}(|h\dot{X}_i|) \rceil$ is more than ten times inferior to its upper bound for any $i \in [\![1, n]\!]$, the integration step h is increased by the same factor 10 to adapt to the new situation that exposes fewer risks of sudden unanticipated evolution.

It is to be noted that classical adaptive methods would also adapt the integration step h to the acceleration values $\ddot{X} = \frac{d^2X}{dt^2}$ that encompass the speed of variation of the derivative and therefore the sharpness of a movement. However, our experiments quickly showed us that the computation of the zeros and small values of the 2^{nd} order derivatives of classic *ODE*s pose important complexity and solvability problems.

Consequently, we did not develop this direction, which could not be generalized to the study of most CPSs.

7.2.3 . Proximity planning

The second adaptive aspect we use in our approach is the adaptation of the integration step to the distance to the nullclines. Once again, the qualitative zones introduced in chapter 5 play a central role. This time, the interesting isoclines constants are not the high values but the smaller ones, describing a physical proximity of the system state with an upcoming qualitative event. We aim to avoid missing any transition between behavioral phases or even between two operating modes. We also would like to avoid 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 associated with small distance values ϵ. As the different qualitative events do not have the same importance (a discrete transition will generally be considered more critical than a change of qualitative state), each qualitative zone will be given a criticality factor $k \in \mathbb{N}$. The more critical the associated qualitative event and the smaller the distance characterizing the qualitative zone, the highest k will be. The value k_{ref} corresponding to the criticality factor of a situation outside of any qualitative zone defined to delimit direct proximity with a qualitative event is fixed at 1. Then, the current value of k will have to be related to the integration step to define the adaptation function. Depending on the policy of the simulation and the favored criteria between number of steps and precision, the relation can be either geometric ($h = \frac{h_0}{k}$) or logarithmic ($h = \frac{h_0}{10^{k-1}}$). In order to stick with the chosen time scale, the second option seems more interesting, but the reduction of the time step will be sharper.

We chose to use a magnitude of 10 for the same reason we evoked in the precedent paragraph. This reduction of the integration step appears in areas that are close to nullclines.

It is to note that this adaptation is to be combined with the reduction to the derivative exposed in the precedent paragraph: the intersection between two qualitative zones defining respectively a high derivative $\dot{X}_i$ and the proximity to a qualitative border $|p_k| < \epsilon$ is even more at risk because the strong value of $\dot{X}_i$ could lead to cross the border supported by $p_k = 0$ faster than in other conditions.

If, at the same time t, X is also positioned in a qualitative zone corresponding to a high value of $\dot{X}_j, j \neq i$, the strong variation generated by X_j will increase the probability of a sudden crossing of the nullcline $\dot{X}_i = 0$.

Such a situation leads to a double reduction of the integration step in order to perfectly situate the crossing time, which is particularly at risk of happening fast.

Let us consider the derivative vector $\overrightarrow{\dot{X}} = \sum_{i=1}^{n} \dot{X}_i \overrightarrow{dX_i}$. If there exists $i \in [\![1, n]\!]$ such that $\dot{X}_i$ changes its sign at a time t, by continuity and supposing the qualitative pilot correctly adapted the integration step h to the value of the highest derivatives, $\dot{X}_i(t-h)$ and $\dot{X}_i(t)$ will be both in a close neighborhood of the nullcline $\dot{X}_i = 0$ represented by the qualitative zones supported by the equation $|\dot{X}_i| < \epsilon$.

The controller is then more careful about the occurrence of mistakes or unplanned behavior in these situations. The adaptation to the proximity of an event can be compared to the adaptation of someone walking near a cliff on a mountain path: this person will walk more slowly and more carefully when close to the cliff in order to avoid any bad move. The reduction of the integration step in a simulation can prevent the appearance of false positives (i.e., computed transitions that do not exist in the system) and detect with precision the transition timing without requiring rollback [39].

7.2.4 . Variable quantization

Quantization can be defined as the process of discretizing a value or a set of values from a continuous space to a discrete representation. Just as we did with the integration step, we assumed that we could modify the quantization precision and adapt it to the qualitative position of the system to deal with the precision requirements that come with the proximity of a qualitative event and the variation speed.

The introduction of a variable quantization is developed in the works on quantized states system (QSS) simulation [132, 133, 134]: these works introduce the modification of the quantization precision to deal with the important gradient of a system's evolution. In [132], the process aims at representing piece-wise linear functions using piece-wise constant functions by approximating the values of the original function on a discrete quantization grid applied on the system state space. Any continuous function quantized with such a process then becomes a step function, with a distance between two sets that depends on the quantization precision.

The quantization of the state space takes place with the definition of rectangles, which define a partition of the initial space on which each possible value included in the same rectangle will be associated with the same abstract value.

Defining a quantization for the numerical simulation can be efficient in saving computation resources if the quantization precision corresponds to the imperatives of the system and to the current simulation step.

However, any reduction of the integration step will modify the requirements for quantization precision. A smaller time step will imply more regular computations and, therefore, a stronger deviation when the uncertainty caused by the approximation is consequent.

Therefore, the challenge is to define a relation between these two elements under the form $l_t = f(l_i, h_t)$ with l_t the quantization precision at a time t, l_i the initial precision at the beginning of the simulation and h_t the integration step at time t.

Modification in the quantization step is a tough process as improving the computation time is an important stake, but the risk of losing too much information is to be considered. As there is no opportunity to regain knowledge during the simulation, any loss is definitive and must be avoided.

Using the abstraction algorithm developed in chapter 4 and the Lie derivative formula, the qualitative model exhibits the qualitative behavior, which includes all the theoretically possible trajectories throughout the set of computed qualitative states and, therefore, perfectly delimits the normal operation of the system. Implementing this knowledge in a qualitative pilot offers the possibility to adapt the simulation step to the situation in order to avoid errors and unplanned transitions and optimize both computation time and precision.

Using the same logic, 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 away from the real trajectory. A solution is to adapt the quantization to the current time step. For this, we use a function that takes as input the initial state space precision and the current integration step and returns the current quantization. Thinking in terms of order of magnitude, we could 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.

It is to note that just as in [40], we thought about adapting the quantization precision to the contractive or expansive nature of the flow functions: a contraction will allow more uncertainty around the real value and, therefore, have given more margin to use bigger quantization grids.

However, given the difficulty of studying the derivatives of the flow condition in the general case, this possibility was not implemented.

7.2.5 . Results and limits

In this section, we apply our approach to a Brusselator (see Example 1 on page 10) 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.

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 7.3). 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

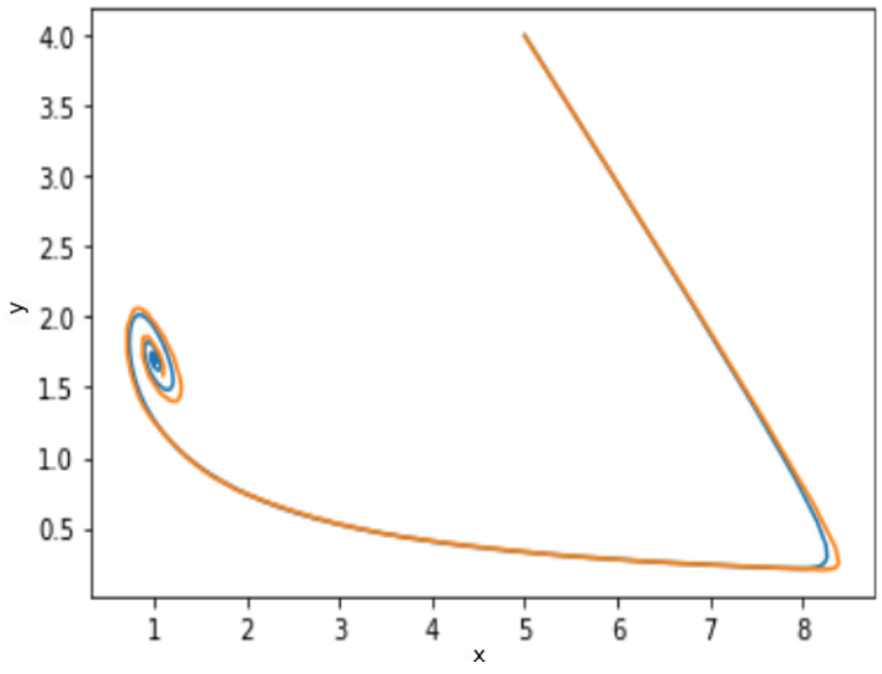

Figure 7.3: Real brusselator simulation behavior (blue) vs. the results obtained with our adaptive simulation (orange)

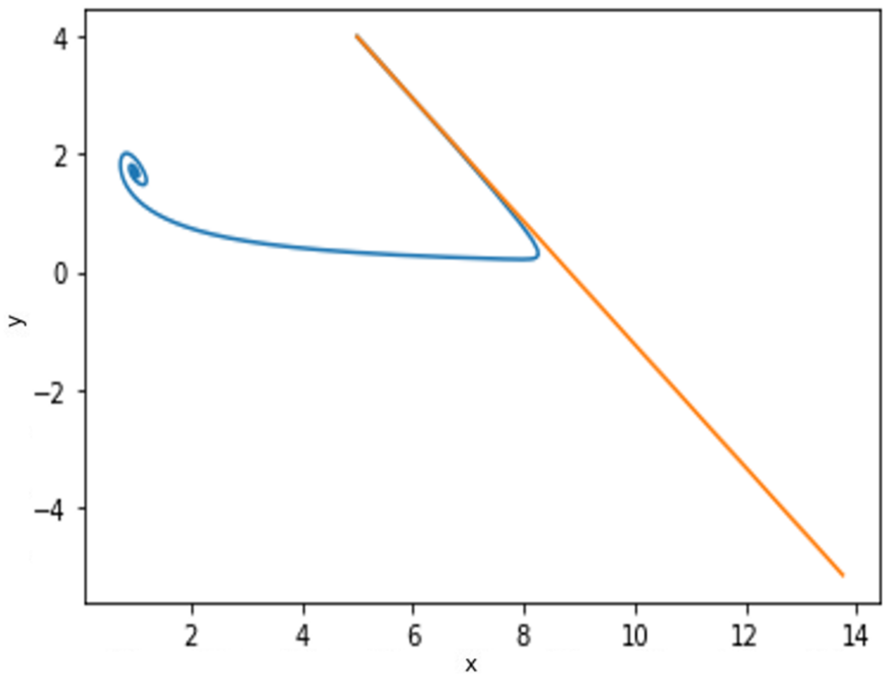

Figure 7.4: Single-step brusselator simulation with $h = 0.1$

equation with a negative value of y after only one step (see Figure 7.4). 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 changes the stopping criterion to a fixed number of steps. The results are shown in Figure 7.5 and Figure 7.6.

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 7.7.

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 the values of x and y. This saves 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 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.

To complete the comparison, we show how false transitions are generated with the fixed integration step method when it gets near the two nullclines. Figure 7.8 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, five false transitions 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.

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.

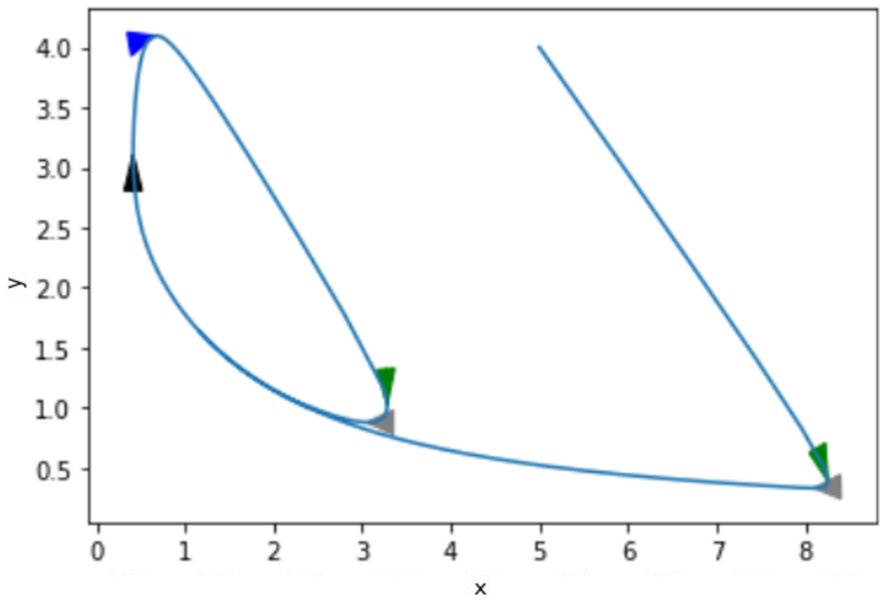

Figure 7.5: Execution of our adaptive algorithm with 280 steps

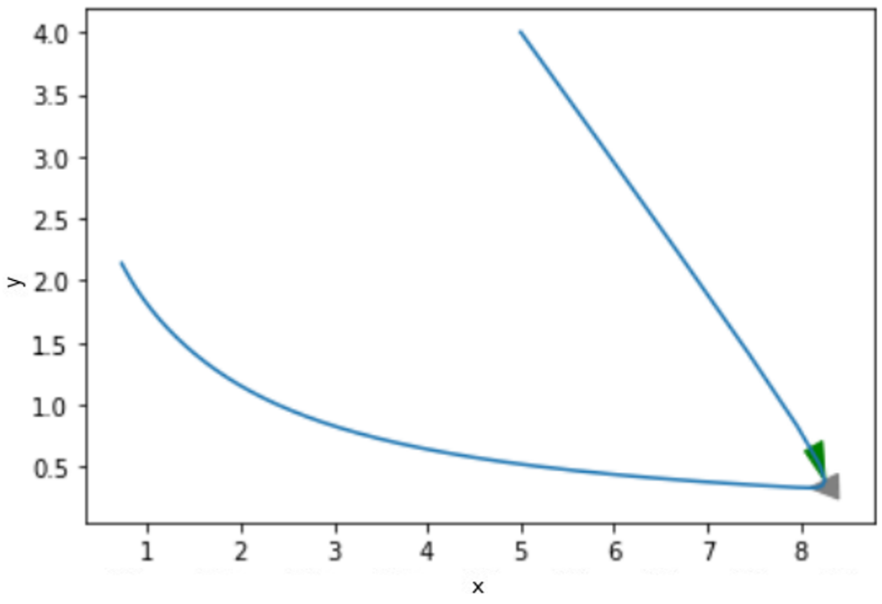

Figure 7.6: Fixed integration step (0.01) with 280 steps

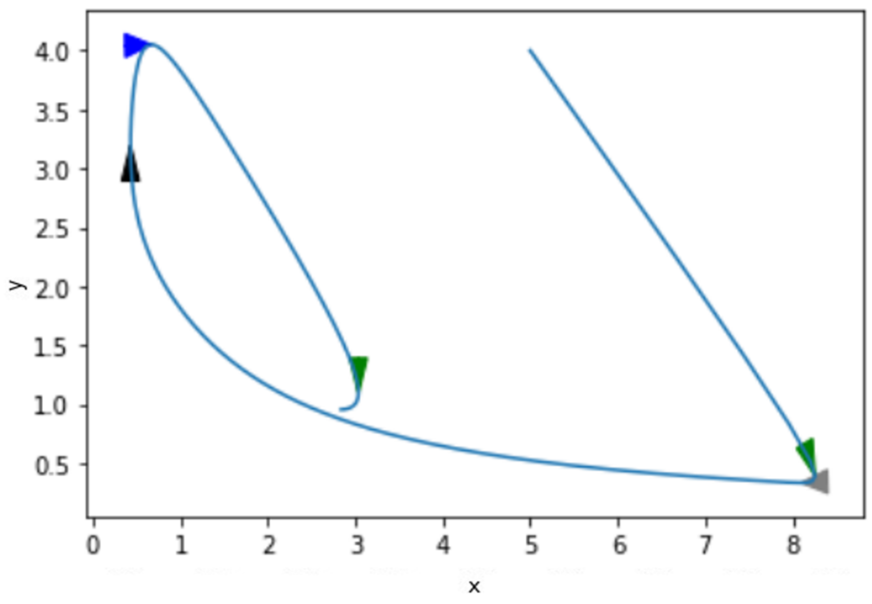

Figure 7.7: Fixed integration step (0.01) with 800 steps.

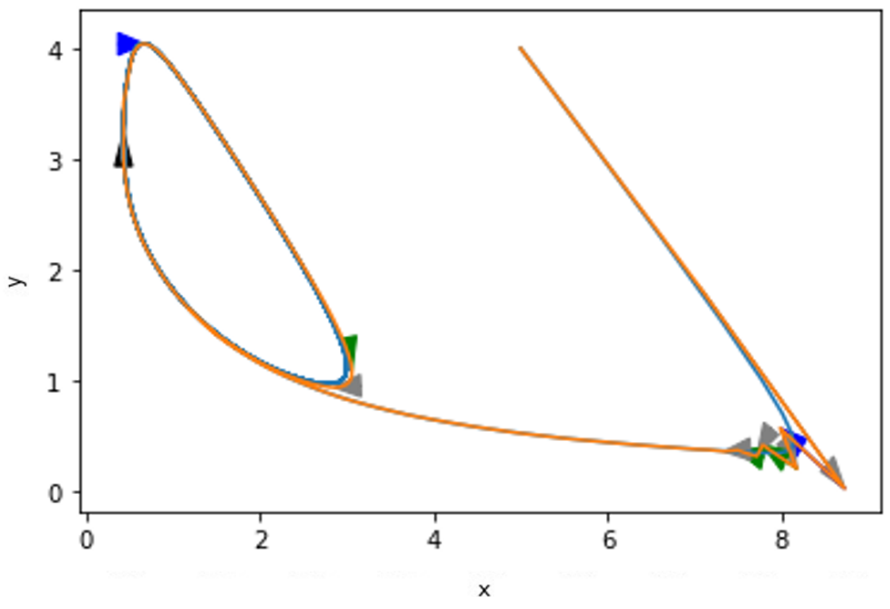

Figure 7.8: Fixed integration step of 0.025 and false transitions

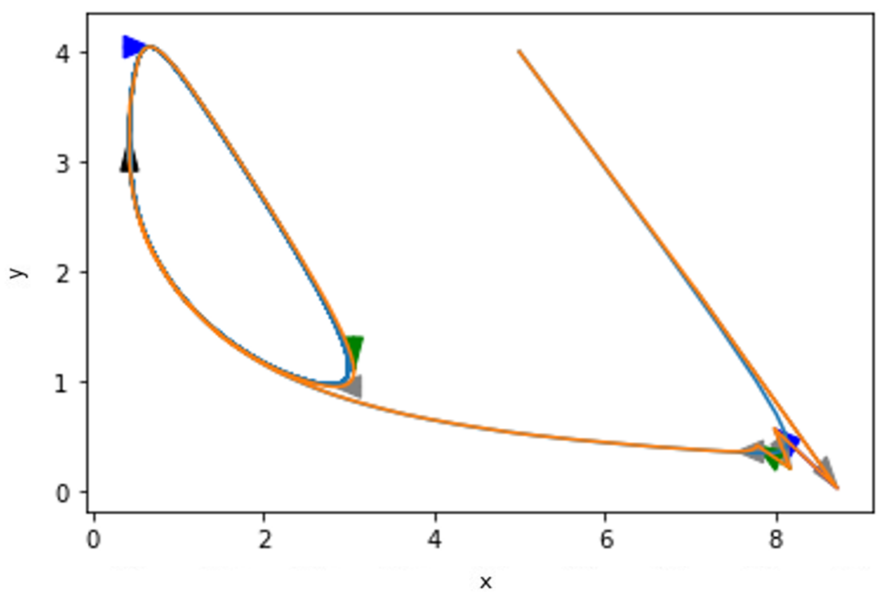

Figure 7.9: Comparison between the reference behavior (blue) and an execution using qualitative event proximity adaptation only.

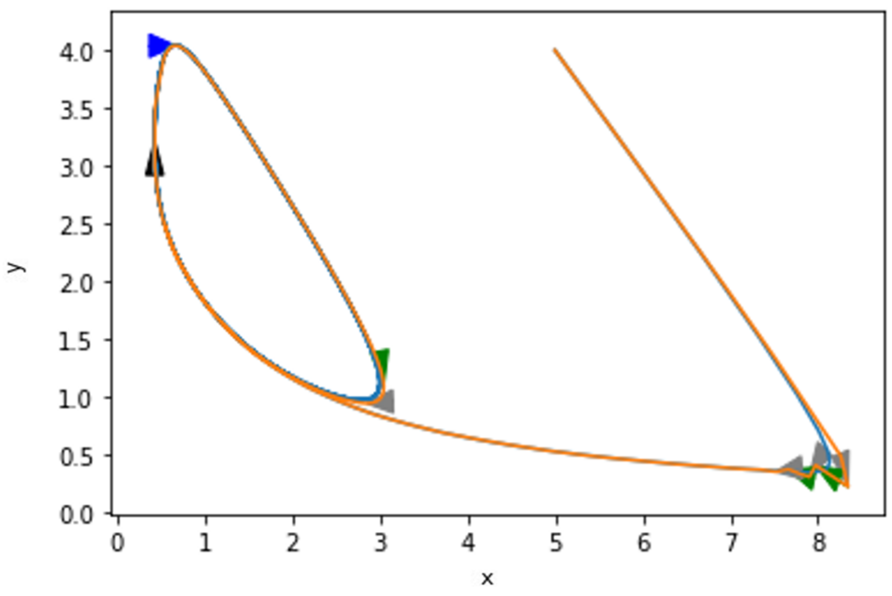

Figure 7.10: Comparison between the reference behavior (blue) and an execution using high derivatives adaptation only.

Figure 7.9 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 7.10 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.

Perspectives and limits

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 moment of the transitions is also more accurate 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 simulation controller automatically reduces the time step in these situations, the inertia is canceled due to the high number of computation points. It 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 that of 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. 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 that a general rule cannot choose for any CPS. We will also have to fully automatize the definition of such a pilot using the qualitative analysis presented above.

Still, some limitations remain that prevent us from generalizing this functionality to any system and fully automatizing the creation of this qualitative pilot.

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. Moreover, it also strongly depends on the case studies, the system's nature, and the stakes around the simulation. All these parameters create an important complexity around the choice of the bounds, and the translation of the system's constraints to a numerical bound is a very difficult task.

The choice of the reference time steps poses the same problem: Is it realistic to choose a generic simulation step that may be used for any type of system in any possible use case? And if not, how do we translate the systemic and situational constraints to an algorithm that could choose an adapted value?

In the next section, we develop a generalization of these ideas in a connected context, which is the monitoring of systems for real-time execution.

7.3 . Real-Time System Monitoring

The different stakes developed in the case of model simulation can be transposed in the situation of system supervision: the need to avoid unexpected transitions, the loss of precision when the state variables have a strong variation with respect to time, and the requirements of improving the execution time without impacting negatively the precision of the operations. The constraint of precision in the occurrence of events is even more important than in simulation, as rollback is not possible in real-time execution. In this situation, the integration step is replaced by the sampling frequency, and the quantization precision is equivalent to the sensor accuracy.

The main difference between the two contexts is that the presence of sensors in a system monitoring situation implies that the loss of knowledge is no longer definitive. If a numerical simulation can generally not claim back information once lost, a monitored system can rely on its sensors to get knowledge back and counterbalance the precision sacrifices to save computation time. This difference completely changes the stakes, as the lack of precision can be solved at any moment depending on the requirements of the situation. Once again, qualitative reasoning has an important role to play in the choice of the sampling instants and frequency and in the required precision of the different measurements.

The areas of interest in the case of system monitoring are mainly the same as in numerical simulations: they are mainly composed of the proximity areas around the nullclines defining either qualitative events or specific use case constraints or landmarks, and areas where one or more variable exhibits a high variation speed in the system's reference. As such events are to be detected before occurrence in order to adapt our expectations of a real-time system, the proximity of a guard frontier is critical knowledge that can be valuable in the case of system monitoring. Moreover, if rollback is considered a last-resort solution for numerical simulation, it is not even an option for real-time system monitoring. Therefore, we proposed to use our qualitative map to adapt the sampling step to the qualitative position of the system.

7.3.1 . Sampling step adjustment

Let $q_p = (m, q_s, q_z)$ be the qualitative position of the system S at a time t with respect to the definition 43. While m gives knowledge about the dynamics of the system and its movements to come, and q_s can be used to explicit the possible future qualitative transitions and, therefore, the current risks at a given time, q_z will inform the operator about which of the identified unwanted event is the most likely to happen at a given point and therefore where the focus should be on the short term. Just as for guided simulation, the interest given to a qualitative zone is then expressed using the same criticality factor k than in section 7.2, whose value will grow with the importance of the considered zone. And just like the simulation step, sampling frequency will increase with the value of k. As each qualitative position is associated with a value of k, it must also be associated with a sampling frequency. If we consider f_{ref} to be the initial sampling frequency and the default frequency when $k = 1$, the sampling frequency f_s will evolve following a function $f(f_{ref}, k)$ that will have to be defined according to the system and monitoring requirements. As in the previous section, the geometric and logarithmic relations were envisaged.

7.3.2 . Measurement quality and state space quantization

As evoked in section 7.2, one major drawback of qualitative reasoning is the loss of information due to the abstraction of numerical values. Once abstracted, the only known method to obtain numerical information from a qualitative space is to use the concretization function [35], which uses the constraints defining a qualitative state to deduce the corresponding numerical values. From this perspective, the case of real-time system monitoring seems to be a perfectly fitting application domain to qualitative reasoning, as the ability to use sensors to gain information offsets the induced knowledge loss. When the quality of the knowledge about the state variable is not sufficient or when one wants to compare the actual trajectory of the system with the expected behavior, one just has to require new measurements from the system sensors in order to refresh and update the numerical knowledge of the system. This possibility is particularly useful when the qualitative behavior tree exhibits a split between two branches and when the prediction of the trajectory is not trivial or ensured. In this situation, the presence of a concretization function β does not make sense, as calling the sensors is enough to get a precise numerical state. Therefore, the choice can be made to adapt measurement accuracy to the value of k of the current qualitative zone. This allows optimization of computation time in state space areas where precision is not required and the focus only on zones corresponding to critical situations. Moreover, one can also modify the quantization precision of the state space in order to adapt the computation complexity and precision, depending on the need for reliable knowledge. This position is somewhat similar to the works on QSS methods [132, 135], where the quantization precision is adapted to the requirements of each simulation step (mainly to deal with flow variation). This point also raises the possibility of using more than one criticality factor and decoupling the precision requirements for the different variables: this implies updating some variables' values more often than others depending on their variation speed and the distance to the nearest events projected in their dimension.

In our experiments, we made the choice to fix the minimal quantization precision to 1% of each value and to upgrade it as much as possible when required. This is a very generic approach, but the choice of a policy will strongly depend on the system's nature and use cases. It is, therefore, impossible for now to highlight a general rule on this point.

7.3.3 . System stopping criteria

If the monitoring of the system is to be automatized, one has to define stopping criteria if something unplanned were to happen.

Such criteria may be both qualitative and quantitative. The first ones are simpler to define as they match our qualitative modeling of the CPS. A qualitative criterion could be the violation of an invariant condition or a change of qualitative state (both represented in our model by qualitative events). Qualitative transitions are stored in the behavior tree as the technically possible invariant violations; such conditions are convenient to anticipate and verify using a qualitative model. To this extent, one can call the abstraction function α_p after every measurement to get the corresponding qualitative position of the measured value. This position must have its validity checked and compared to the abstraction of the previous measurement in order to know if a transition took place. If it indeed happened, the existence of this transition in the behavior tree must be verified.

If the qualitative position or the previous transition is not valid, we can deduce that the system exhibits an invalid behavior and that there must have been a malfunction. Stopping the system to avoid further error propagation is then necessary.

The other category of stopping criteria that can be developed is based on numerical analysis. It consists in a comparison between the last measured state variables values to reference values previously obtained by numerical simulations or by uncertainty propagation methods such as flow-pipe [39, 41]. This intersects with sampling frequency adaptation as the deviations are likely to appear in the state space areas involving sharp movements. The next step is the choice of a maximum authorized error threshold, either expressed by a maximum distance to the expected trajectory or by a limit rectangle that the trajectory should not leave.

7.4 . Application to Design Space Exploration

As explained in the introduction (see section 3.2 on page 43), the first significant element required to process the exploration of the design space is an adapted representation of the model. Using a discretization process, we get a qualitative representation of the system corresponding to $QM = \langle Q, X, B, F, I, T, P\rangle$ with B a mapping from each mode q in Q to a set of qualitative borders defining the qualitative state they separate. The qualitative states are then represented as an array of $\{-1, 0, 1\}$, showing the position of the corresponding state with respect to each frontier in B. This representation can be applied to fully designed systems or more abstract ones with non-valuated parameters in P. The second case implies the impossibility of computing the transitions between qualitative states, as this knowledge requires the computation of the sign of the Lie derivative [35] to deduce the allowed transition directions. For a polynomial p whose zeros define a border of the system, the Lie derivative corresponding to p can be written as

$$L_X(p) = \sum_{X_i \in X} \frac{\partial p}{\partial X_i} \frac{\partial X_i}{\partial t}$$

Determining the sign of this expression requires the ability to fully evaluate p and the derivatives of all the variables X_i.

As DSE requires optimization criteria or necessary constraints, it is possible to integrate these elements in the construction of the frontiers during the abstraction process. If $R(P, X) \in \mathbb{K}^{n+m}$ is a predicate on X and P that should be satisfied by the system, then the equations composing $R(P, X)$ can be considered individually to define new borders.

7.4.1 . Parameters evaluation

The second necessary aspect of DSE is the choice of an analysis method. When a value or a subset of values is available for P, it is essential to test its pertinence using an adapted solver. Many behavioral properties of a dynamic system can be conveniently represented using temporal logic [136]. However, predicate satisfiability for temporal logic is more complex and costly than first-order logic (FOL) evaluation. Moreover, qualitative reasoning gives us an adapted support to reason on first-order logic predicates. We will limit ourselves to the properties expressible as FOL predicates to apply the possibilities of qualitative reasoning. This choice implies representing the behavioral constraint we want to express as predicates.

For example, let us suppose we want a deterministic behavior for the system modeled through its discretization in qualitative states. In that case, we may wish to have a direct transition from the qualitative state (m_1, s_1) to (m_1, s_2). In temporal logic and considering a temporal structure $(\mathbf{T}, <)$, we could write the predicate corresponding to a necessary transition $q_s = (m_1, s_1)\ \mathcal{U}\ q_s = (m_1, s_2)$ with s_1 and s_2 qualitative states of the system in mode m_1, i.e., abstractions of values of X following an abstraction function adapted to the system. In FOL, we could write it using the Lie derivative such that $(m_1, s_1) \implies \forall\, b \in B(m_1, s_1) \setminus B(m_1, s_2),\ L_X(p_b) * p_b(s_1) > 0\ \wedge$ for $b_f \in B(m_1, s_1) \cap B(m_1, s_2),\ L_X(p_{b_f}) * p_{b_f}(s_1) < 0$ where $B(m_1, s_i)$ is the set of borders defining the qualitative state s_i in the mode m_1 and p_b the polynomial function whose zero corresponds to b. This predicate corresponds to the evaluation of the sign of each $L_X(p_b)$ and its comparison to the evolution of p_b after crossing the corresponding qualitative frontier. A qualitative description of the system gives us a convenient toolbox to express the dynamic logic of the system as first-order logic predicates.

In our case, we keep using *Z3* as it perfectly handles property verification on parameter values. We use it as a *Python* library and create a solver s to which we add the constraints on the system's design, as will be shown later in the algorithms we use for DSE. The command $s.check()$ returns the satisfiability of the created problem under the given constraints. Compared to constraint solving on the state space, the main difficulty here is that there must be many calls to the solver to fix a value or a set of values of P that verify the constraints.

7.4.2 . Design space exploration

The last element of an efficient DSE is an optimized research method to choose the values or the sets of values of P. As already mentioned, we assume that the design space is bounded. To develop a research method, we have to separate two prominent cases depending on the nature of the design space.

Finite Design Space

In the literature, DSE methods are mainly adapted to optimize the search for solutions in finite design spaces. Many practices have emerged, including genetic algorithms [137], machine learning methods [138], and other discrete optimization methods. We chose to exploit the permissiveness of the SMT solver *Z3* to use branch-and-bound to specify the different parameters of P sequentially without dealing with every possibility. In branch-and-bound, we compute for each parameter p_i of P the satisfiability of the constraints set for every proposed value $v_{i,j}$ associated with every satisfiable combination of $v_{k,l}$ for $k < i$. Here, $v_{i,j}$ corresponds to the j^{th} authorised value of the i^{th} element of P. By cutting the branch of the non-satisfiable partial combinations of P_{1-i}, we save the computation time that could be lost to test trivially impossible solutions. This algorithm is detailed in Algorithm 3, where *possibleConfig* retains the authorized partial evaluation, allowing the constraints *Constr* to be satisfiable. The function *toConstr* transforms the instantiation of the current parameter p_i to a constraint expressed as an equality. This new constraint is noted *newConstr*, and its conjunction with *Constr* is noted *compConstr*. $\mathbf{D}_P[p_i]$ contains all the authorized values$v_{i,j}$ of the component p_i of P. The operation $pre + v$ corresponds to the association of the authorized partial configuration pre of the $p_k,\ k < i$ with the considered value v of p_i.

```
Data : $P,\ \mathbf{D}_P,\ Constr$
Result : Possible values of the parameters in $P$
$possibleConfig = [[]] * length(P)$
for $i \in (0, length(P) - 1)$ do
    $p = P[i]$
    for $pre \in possibleConfig$ do
        for $v \in \mathbf{D}_P[p]$ do
            $newConstrs = toConstr(pre + v)$
            $compConstr = newConstr \wedge Constr$
            $s = SMTsolver$
            $s.addConstraints(compConstr)$
            $an = s.Check()$
            if *an* == *SAT* then
                $possibleConfig[i + 1].append($
                    $pre + v)$
                $)$
            end
            if $isEmpty(possibleConfig[i + 1])$
              $\wedge\ i < length(P) - 1$ then
                Stop
            end
        end
    end
end
return *possibleConfig*
```

Algorithm 3 : Branch-and-Bound algorithm.

Infinite Design Space

Although the exploration of a discrete design space is often practiced on finite discrete sets, developing the same techniques for continuous or infinite research spaces can be crucial. We develop an approach based on dichotomy, i.e., a successive partition of the search space into two distinct parts. For a single parameter p, we define a minimum width of the remaining definition set, and we subdivide it until we find that there is no solution in the set or that the current width of the set is smaller than the minimum width. The algorithm is shown in Algorithm 4, where $DichotomialCut(S_p)$ proceeds to a split of the set S_p. In the general case, if S_p is an interval on $\mathbb{R}$, noted $[a, b]$, the separation is completed by computing the center of the interval $c = \frac{a+b}{2}$. Otherwise, if S_p is composed of many convex non-contiguous sets, the separation is done by splitting S_p between two non-contiguous components. $minSize$ corresponds to the criteria on $max(\|a - b\|)_{a,b \in set}$ to stop the splitting process. As in Algorithm 3, $Constr$ represents the set of constraints on the design parameters. $allowedSets$ contains all the explored sets and Boolean values corresponding to the satisfiability of the design constraint on each of these sets.

To deal with multi-dimensional P, we can combine Algorithm 4 and Algorithm 3 to use dichotomy on every $p_i \in P$ sequentially and to divide the different combinations with a disjunction of cases.

Interestingly, we process once again to a discretization of a continuous space just as in the state space partitioning. The obtained partition of $\mathbf{D}_P$ is based on the respect of the design constraints. Using an optimization function could also give more tools to refine such a partition based on its magnitude and variation. Moreover, the computation of its derivative would allow us to compute its critical points and, therefore, its maximums.

Non-Uniform Probabilistic Distribution

In the main part of the cases, $\mathbf{D}_P$ is a subset of $\mathbb{R}$ and often takes the form of an interval. However, in some situations, it may appear that the design space does not take this form. For example, if prior knowledge is available regarding the probability of the different possible values, the set can no longer be represented using a simple interval. Some works, such as [139], proposed solutions using probability density functions to represent this irregular distribution. This can be seen as a generalization of the non-probabilistic case considering that $\forall (a, b) \in \mathbb{R}^2$ such that $a < b$, the interval $[a, b]$ can be represented by a probability distribution following the uniform probability density law $U(a, b)$ on the same interval.

One can also represent discrete sets using discrete probability laws. Therefore, it is possible to model many design spaces using probability functions. Using the expression of the probability density function for each $p \in P$, it is

```
**Data :** $p,\ S_p,\ Constr,\ minSize$
**Result :** Domain of possible values for $p$
$allowedSets = []$
$S_b,\ S_u = DichotomialCut(S_p)$
for $set \in \{S_b, S_u\}$ do
    $newConstr = toConstr(p \in set)$
    $ConstrTot = newConstr \wedge Constr$
    $s = SMTsolver$
    $s.addConstraint(ConstrTot)$
    $an = s.Check()$
    if $an == Unsat$ then
        $allowedSets.append((set, False))$
    else
        if $max(abs(a - b))_{a,b \in set} < minSize$ then
            $allowedSets.append((set, True))$
        else
            $allowedSets.concatenate($
                $Dichotomial\ search($
                    $p, set, Constr, minSize$
                $)$
            $)$
        end
    end
end
return allowedSets
```

Algorithm 4 : Dichotomial search algorithm

possible to pick values of P according to these density functions and combine this biased selection with the previously presented methods. However, this way of exploring the design space better fits the classical numerical simulation verification as it also applies gradient descent to orient the choice of new values for the next simulation. It constrains the differentiability of the utility function according to the researched element. This also corresponds better to the selection of the variable's initial values than to parameter valuation.

7.4.3 . Experimentation

At this stage of our research, we have not been able to apply the presented algorithm to concrete cases, and we did not set up a test benchmark either. We planned to test these algorithms on both simple systems such as a Brusselator system, with positive parameters a and b and state variables x and y, whose dynamics is given by Equation 2.1, and on more complex ones such as the Lorentz system, with positive parameters σ, ρ and β and state variables x, y, and z, whose dynamics are given by Equation 7.4.

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases} \tag{7.4}$$

Both systems exhibit different behaviors depending on the value of their parameters. The convergence or periodicity of such systems is an essential element in the trace of a system, so we will try to express the sets of validity of the convergence of each one for finite and continuous sets of allowed values. If our method allows us to obtain sufficiently precise results corresponding to theoretical calculations, it will be possible to consider further analysis and computation to improve the efficiency and the generality of the process. However, to consider the result interesting, it will be necessary to measure the computation time of our tests to ensure that there is either a gain in complexity, permissiveness, or generalization.

7.4.4 . Perspectives and future works

The next step to perfectly combine our different subjects would be to express the properties of the state space and the solution of the various constraints on the variable X depending on the chosen value of the parameters P. The validity of a predicate would be expressed as $X = f(P)$ for the subset of $\mathbf{D}_P$ that allows these predicates to have a solution. To achieve this goal, we searched many different methods and looked at other tools, such as modal solvers or Wolfram Alpha, which give encouraging solutions in that direction. However, it does not generalize well for systems with too many variables. Other tools that have been studied allowed this type of resolution but only for low-degree polynomial functions. There is no possible generalization for now.

8 - GENERALISATION

As our works follow and complete works focusing on polynomial systems, the proposed abstraction methods were designed mainly for systems with polynomial constraints and dynamic equations. This is due to the simplicity of polynomial equations resolution until a certain degree, while more general functions do not offer such convenience for resolution. As polynomials are almost the simplest mathematical functions to manipulate, the framework of the polynomial system is very advantageous but still quite limited. It is, however, possible to generalize this work to other categories of functions, such as rational functions.

Depending on the nature of the considered equations, the abstraction process will be different in order to fit the technical constraints of the state space discretization process.

8.1 . Rational Functions

Considering that every rational function can be written in the form $\frac{p_1}{p_2}$ with p_1 and p_2 two polynomials, the set of all rational functions on the variable set X and written $\mathbb{K}(X)$ can, by definition, be expressed as a product of set $\mathbb{K}[X]*$ $(\mathbb{K}[X]^*)$. Therefore, computing the zeros of a rational function is equivalent to computing the zeros of its numerator polynomial p_1 and verifying that they do not match with the zeros of its denominator p_2. This implies that *Sympy* and *Z3* solvers, primarily thought for polynomial functions, can perfectly deal with rational equations.

Moreover, as it is easy to prove that $\mathbb{K}(X)$ is closed under the derivation just like $\mathbb{K}[X]$, manipulating rational functions for state space abstraction does not change the situation for the computation. It makes the resolution more complex and the execution longer. However, as computing the qualitative map is an offline operation, the computation time does not constitute an important criterion. The qualitative model is supposed to be generated before the execution of the processes (be it a simulation or the monitoring of the system) that use it.

The main problem will be the apparition of a singularity area near the poles of the fractions, which will cause the qualitative borders to have unpredictable shapes.

In spite of this inherent complexity, the generalization of our process to rational fractions does not pose major problems.

8.2 . Qualitative Tendencies and Hybridization

One of the limits of our state space abstraction process is that it strongly depends on the ability of the symbolic solvers to find the zeros of the system's polynomial equations. Consequently, it can exhibit difficulties when in the presence of high-degree and multivariate polynomial equations. A solution to circumvent this problem would be to apply a new abstraction process of the most complex equations to decompose them into local and simpler expressions that would still exhibit the same qualitative characteristics but with a smaller complexity.

Our objective is to extend the notion of OG to local rectangles to offer a qualitative abstraction of the functions of a system using piecewise decomposition. To this end, we will combine the tools presented earlier to propose a qualitative local categorization and simplification of the equations of a model. We will then reduce the complexity and make the model's behavior easier to understand at the price of a loss in precision.

This function abstraction based on a state-space subdivision is strongly inspired by the concept of hybridization [140]. In our case, we propose to create new operating modes for each sub-space of the state space of a mode depending on the dominant term (which we call qualitative tendency) in order to consider only the most influential terms in every area to deal with more complex expressions.

Let us consider the evolution of a variable y that varies according to the function $y = (t-3)^3 - 4 * t^2 - 2 * t$. From a global point of view, the observed behavior of the variable will be that of a classic cubic function, with two phases of polynomial growth separated by a decay phase. On a small scale, this decay period is essential to determine a stop in the increasing qualitative behavior observed earlier. However, on a larger scale, if we are just interested to know the difference of magnitude at two times t_1 and t_2, this decrease period has no interest: the tendency tells us that the variable followed a cubic growth between t_1 and t_2. This difference between general behavior and local trajectory is the center of this contribution. Depending on the level of granularity and the information we seek, we can adapt the expression of the equations of a system to optimize its study without altering the results.

8.2.1 . Dominant terms and qualitative tendencies

As exposed in section 4.3, complex polynomials can become a problem if the capacity of the solver is exceeded. This can happen when X is composed of many variables and if the treated polynomials are of high degree. To keep these case studies in the range of qualitative reasoning, we worked on a function abstraction process [141] based on order of magnitude [111] and OG reasoning. After having defined a reference value $v_{i,m}$ for every component $X_i \in X$ and every mode $m \in \mathbf{Q}$ expressing the anticipated order of magnitude of the variable X_i in the mode m and set of negligibility criteria $\nu_{i,m}$, the process applies a hybridization method [93, 140] to abstract each function considered as too complex in a piece-wise continuous simplified function, keeping for each sub-division of a mode state space the terms considered as dominants and suppressing the terms whose impact on the dynamics are considered as locally negligible. Hybridization is often used to simplify continuous complex systems by transforming them into a hybrid system that reproduces their properties and behavior. The idea of hybridization consists in approximating a too-complex continuous vector field by a hybrid system with a collection of simpler (constant, affine, or even polynomial) vector fields [140]. In the case of already hybrid systems, it would lead to splitting each mode of the system into several modes with simpler equations. The comparison between the different terms of an equation is possible using OMR [113]. More precisely, as OMR can be either absolute or relative [113], we especially use relative OMR to perform a comparison between the relative magnitude of an equation. Absolute OMR compares different variables to a set of fixed reference values, while relative OMR compares variables with each other as long as they are comparable. The different comparison systems use their own operators, but the operators Neg (Negligible) and Vo (Neighbor) seem to make a consensus. We also chose to use Co (Comparable) from *Rom*.

Relative OMR and OG reasoning give a good basis for abstracting the functions defining the CPS. In the case of high-degree polynomials, equivalent classes impose the monomials of different OG to have different behaviors to infinity. Consequently, monomial terms of different degrees applying on the same variable of X or parameter can be simplified if the considered variable reaches sufficiently high values. The chosen negligibility criterion associated with the variable gives the threshold authorizing this abstraction.

The protocol to apply relative OMR and OG reasoning for polynomial equations is as follows.

After separating each of the target equations into monomial terms, the algorithm will discretize the state space based on the relative prevalence of each of the monomial terms on the others to create a piecewise continuous function constituted of sub-functions simpler than the original one. Each sub-function will correspond to a new mode of the qualitative model, meaning that hybridization is applied to each mode separately. The abstraction of the original function in each of these newly created modes is called **qualitative tendency** of the function (see definition 44). It increases the number of operating modes of the qualitative model and the computation capacity for each.

Definition 44 (Qualitative tendencies) If $f : \mathbb{K}^n \to \mathbb{K}$ is a function on $\mathbb{K}$, let f_r be the piece-wise continuous function abstracting f obtained according to the previously presented instructions for negligibility criteria ν_{X_i} for each variable X_i. We call qualitative tendencies of f the continuous functions f_i defined on a rectangle of $\mathbb{K}^n$ composing a continuous section of f.

Qualitative tendencies are thus defined as the sections of the abstracted formulation of the equations of the system. It corresponds to a subspace of the state space where the respective orders of magnitude of the variables and where the local magnitude or growth of the equation implies a specific influence relationship between its different terms. In the case of equations involving different variables, the homogeneous terms are compared to each other as the presence of an addition supposes a physical homogeneity in the added terms. As example, consider a function $f : x \mapsto x^3 - 2x^2 - 2$ defined on $\mathbb{R}$. If the negligibility criterion is chosen to be of 0.1 (meaning that $x\, Ne\, y$ if $\frac{|x|}{|y|} < 0.1$), then the obtained abstracted piecewise continuous function can be expressed as

$$f_p = \begin{cases} -2 \text{ if } |x| \leqslant \frac{1}{\sqrt{10}} & x^3 \text{ and } x^2\, Ne\, 2 \\ -2x^2 - 2 \text{ if } \frac{1}{\sqrt{10}} \leqslant |x| \leqslant \frac{1}{\sqrt[3]{10}} & x^3\, Ne\, 2 \text{ and } x^2 \\ x^3 - 2x^2 - 2 \text{ if } \frac{1}{\sqrt[3]{10}} \leqslant |x| \leqslant \sqrt{10} & \\ x^3 - 2x^2 \text{ if } \sqrt{10} \leqslant |x| \leqslant 10 & 2\, Ne\, x^3 \text{ and } x^2 \\ x^3 \text{ if } |x| > 10 & x^2 \text{ and } 2\, Ne\, x^3 \end{cases}$$

It is also possible to slightly change this expression to make the abstracted function continuous on $\mathbb{R}$ by evaluating the suppressed terms on the discontinuity points when the continuity of the abstraction is a constraint. This modification requires a few more computations but does not radically change the process or the result.

This way, it becomes possible to create another qualitative model of the same system, which will contain less precise and more artificial frontiers, but it will guarantee the possibility of processing and refining the model better than with high-degree polynomial equations.

8.2.2 . State space sub-partitioning

The point is now to apply this partition based on OG to study the different equations of the model and to map their behavior in the state space. As mentioned earlier, the approximated behavior of a variable depends on the scale and the time of observation. To know which abstraction will be authorized, we must first confront the equations to the magnitude taken by the variable in the state space and then study the time slot we are interested in. This will require a mapping of the state space that associates each set of the partition to the dominant terms to be kept in the approximated equation. To take our first example back, the dominant term of the function $x \mapsto x^2 - x + 1$ will not be the same depending on the current value of x. If we choose a negligibility criterion of 10%, then the dominant term on $(-\infty, 10]$ and on $[10, +\infty)$ will be x^2, the dominant one on $[-0.1, 0.1]$ will be 1, while there will not be a single dominant term on $[-10, 10] \setminus [-0.1, 0.1]$.

We achieve this mapping by analyzing the different terms of each equation and its variables, giving importance to the different terms in each area of the variation space. Theoretically, it is possible to extend this manipulation to logarithmic-exponential functions. However, for practical reasons, we focussed our tests on polynomials only.

First, we explore the state space by analyzing the definition space of each variable to avoid making useless computations. For example, we know for a closed chemical reaction that the total quantity of one component will not exceed the sum of the initial quantity of all the components and that its density will not exceed 1 by the definition of this variable.

Using the system's equations and negligibility criteria, it is possible to automatize the model simplification with symbolic rules (particularly regarding the degree of the monomials of a polynomial equation). Polynomials can be studied using solvers to compute their zeros. Each term must be studied separately and compared to the rest of the function using the negligibility criteria and the scale chosen for this dimension of the system. This comparison will imply their ranking between known landmarks where their local orders of growth converge.

Each monomial function has an OG equal to its order. The resolution of the equalities between the monomial terms in a function is necessary to anticipate their dominance inversion.

The symbolic rules will have to be adapted to the negligibility limit. However, we can express them as: $f(x) = o(e^{g(x)})$ for every f and g two polynomial functions if $x > v_x$, with v_x the characteristic unitary value of the variable x (corresponding to the most suited magnitude for the values taken in the system), or $m(x) = o(n(x))$ if m and n are two monomial functions with $c(n) - c(m) = k > 0$ and $x > \sqrt{v_x}^k$.

In the case of multi-variable functions, we first require a factorization tool to separate the variables in the expression of the function. Once this step is accomplished, we treat the univariate monomials just like in a classic polynomial, treat the polynomial factors as if they were independent, and apply the simplification in the complete function. When a product of many factors cannot be simplified, we prefer not to simplify it because of the many influences it merges. The result is a product of simplified polynomials that we consider our final function.

The landmarks of the partition are placed where there is equality between different components of the same equation. Around these landmarks, we place an area of qualitative equality, where even if the components are not equal, they are considered comparable, being too close to neglect one.

Therefore, for a function f composed of n monomials f_i, we study separately the f_i one after another. We inject them in a target function f_p initialized at 0 such that we compute and solve the inequalities $|f_i| < \frac{|f_p|}{v_f}$ and $|f_i| > |f_p| * v_f$. In the intervals obtained by computation of these inequalities, we consider respectively that f_i is negligible compared to f_p (meaning that in this interval, the expression of f_p will not evolve) and that f_i dominates f_p, (in this case, we then impose on these intervals that $f_p = f_i$). On the rest of the state space, we add f_i to f_p as they are comparable.

8.2.3 . Property preservation

As already mentioned, any model and, therefore, abstraction of any system cannot preserve perfectly every property of the initial system [2]. The abstraction of high-degree polynomials will, therefore, imply a loss of knowledge and precision in the representation of the system's properties. If an equation defining an important aspect of a system is to be abstracted using qualitative tendencies, one must first decide which of the initial equation's characteristics must be maintained in the abstract model.

The challenge would then be the translation from this requirement (expressed in natural language or with logical and mathematical predicates).

As the state space abstraction process presented in chapter 4, following the equation abstraction of this section, uses the polynomials to compute their zeros and the zeros of their derivatives, an accepting condition to abstract the polynomials would be the preservation of these zeros and of the extrema (corresponding to the zero of a derivative) of each equation so that the transition conditions would be exact and would not cause a gap between the original and the approximate nullclines. The preservation of the zeros and extreme points is a real stake because all the qualitative characterization of numerical values is based on the sign of the systems equations and of their successive derivatives. If the sign of some of these equations is miscomputed because of their abstraction, the resulting qualitative state will be wrong. If a numerical error can be tolerated if limited to a decided extent, a false qualitative abstraction has more harmful consequences. A change in one digit of the qualitative state expression can invalidate a major predicate of the system behavior and completely destroy the coherence of the reasoning or of the simulation.

In order to preserve the zeros and extrema, the abstraction process must avoid any under-approximation in the areas where the original function may change its sign or meet an extremum. Consequently, if the studied function f can be expressed as a sum of monomials $f = \sum_{i=0}^{k} f_i$, the abstraction process should authorize a simplification on the area E_a iff there exists a unique $i \in [\![0, k]\!]$ such that $\forall\, x \in E_a,\ \forall\, j \neq i,\ |f_j(x)| \ll |f_i(x)|$. If there are multiple dominant terms that are neighbors or even comparable, the order of magnitude of a sum of such terms is not determinable.

Consequently, the existence of multiple dominant terms in a single area stops the simplification process in this subspace, as the sum of two terms whose orders of magnitude are close cannot be positioned and could be negligible to other terms of the initial equation. This situation can happen in multivariate polynomials, such as $f = 3x_0^2 - 2x_1^4 - x_0$. In the subspace defined by the inequalities $\sqrt{\nu_f}x_1^2 < |x_0| < \frac{x_1^2}{\sqrt{\nu_f}}$, the two monomials of highest degrees are comparable, and the term x_0 cannot be neglected because its value could change the sign of f if the monomials of same magnitude have a different sign. Otherwise, adding the study of the sign of each monomial can be considered, but this possibility is limited to the systems taking values in $\mathbb{R}^n$. Moreover, this functionality would clearly increase the complexity of the abstraction process as both magnitude, sign, and operators will have to be studied, which will explode the number of cases and new modes emerging from the hybridization.

8.2.4 . Bounding the error

In this section, we present theoretical justifications that the obtained approximations give an error that we can confine or prove as negligible compared to the function itself. For zero order functions expressed as $f = g + h$ with h a term or a sum of terms negligible compared to g on an interval $[a, b]$, the error between f and its approximation $f_p = g$ is $|h|$, that we supposed negligible. If we are interested in first order dynamics of f, then let us call $F = \int_a^b f(x)dx + F(a) = F_x + F(a)$, and use the same notation for G and H. If h is negligible compared to g on $[a, b]$, then we can affirm that, using the intermediate value theorem, g will not change its sign. As $h \ll g$, if the negligibility criteria is ν, we have $|h| < \frac{|g|}{\nu}$. Therefore,

$$\forall t > a, |\int_a^t h(x)dx| < \int_a^t |h(x)|dx < \frac{\int_a^t |g(x)|dx}{\nu} = \frac{|\int_a^t g(x)dx|}{\nu}$$

This means that $|H_x| < \frac{|G_x|}{\nu} \implies H_x \ll G_x$. This way, we can propagate the negligibility to first-order dynamics and, with a simple recurrence, to higher-order ones. Moreover, with a simple property, we can affirm that

$$\int_a^t |h(x)|dx < (t-a) * max_{y \in [a,b]}(|h(y)|)$$

which means that we can easily bound the final error on finite rectangles.

In the case of infinite intervals, the first part of the proof still holds. However, it is not possible anymore to limit the error as the function may not have an extremum on it.

In the presence of non-rational transcendental functions such as exponential, logarithmic, or trigonometric expressions, the ability of polynomial solvers to symbolically find the zeros of the equations does not hold anymore. The previous propositions and processes do not apply to systems defined at least partially by such equations. Therefore, abstracting the expressions before constructing a qualitative model is a necessity. Depending on the nature of these functions, the abstraction possibilities will show various benefits and precision.

8.3 . Periodic and Stochastic Functions

The previous paragraph focused on the abstraction of polynomial functions, but many systems exist that cannot be represented by polynomial functions alone. Some of them imply periodic or stochastic terms in their expressions, such as the SHSs. Our work on the OG and hybridization only applies to polynomial functions, but we also propose other simplifications adapted to other categories of functions, taking advantage of properties of periodicity and randomness.

In the case of dynamic differential equations on long intervals, it is more complicated to assume that we can neglect a term of the equation. Even if a term is relatively negligible with respect to the main part of the equation, it can still be necessary if the simulation duration is long enough because the deviations will add up. As $\mathbb{R}$ is archimedean, adding too many terms that are negligible compared to x^n may give a result that is not negligible compared to x^n. Consequently, reasoning dynamically on orders of magnitude in long time intervals has limitations. Still, some specific functions, such as periodic ones, exhibit properties that can still allow us to make abstractions and simplifications depending on their amplitude and frequency. If the system is represented by an equation $f = g + h$ with g non-periodic and h periodic, we aim to know if it is possible to suppress or to replace h in the simplified expression f_p of f.

Let us consider the interval of study I_T of length T. The frequency of h will show if h is locally periodic, *i.e.* if we can still observe the periodic nature of h when we consider only its local behavior on I_T. When the period p of h is larger or comparable to I_T, then we can consider h as non-periodic, and the reasoning of this paragraph does not apply. Otherwise, if $p \ll T$, the question is whether it is possible to use this periodic nature to simplify the expression. By definition, a periodic function varies in cycles around its mean value m, with higher and lower values that will alternate. Our logic is to consider that the lower and higher values may offset each other with enough periods in an interval. To ensure that, we must first compute m on an entire period to locate its value compared to the characteristic value v_q of the associated variable (corresponding to the order of magnitude of the values taken by the variable on which the equation applies). Let us make the hypothesis that every function we deal with is at least piece-wise continuous and, therefore, locally integrable. We can compute the mean value taken by the function on one period with

$$\bar{h} = \frac{\int_t^{t+p} h(x)dx}{p}$$

If $m \ll v_q$ according to the chosen negligibility criterion, with v_q the characteristic value of f, then the mean effect of h on the complete function can be neglected (by definition of the negligibility). However, we still have to look for the volatility of h, represented by its amplitude. With h_M and h_m the extreme values of h on one period, we still have to ensure that $|h_M - \bar{h}|$ and $|h_m - \bar{h}|$ are still not too important with regard to v_q. If $|h_M - \bar{h}| < v_q$ and $|h_m - \bar{h}| < v_q$, the volatility of h is contained and our last criterion is then satisfied. A high amplitude compared to v_q could provoke qualitatively visible behavioral changes as high volatility generates an important impact on processes with a limited mean. If $|h_M - \bar{h}| < v_q$ and $|h_m - \bar{h}| < v_q$ but $\bar{h}$ is not negligible compared to v_q, we can still propose a simplification of h by replacing it by average value $\bar{h}$. If the movement around $\bar{h}$ is sufficiently limited, it is possible to consider only the mean value to identify the qualitative tendency.

When h is applied to a zero-order equation, the compensation property comes easily from the periodic nature of the function and the required predicates. This means that the deviation from the mean is restricted by a value supposed to be at most comparable to v_q. In the case of higher-order differential equations, the demonstration is based on the Fourier series of periodic functions. If the function h is periodic on I_T, then it can be expressed as

$$h(t) = \sum_{n=\infty}^{+\infty} a_n(h) e^{-2i\pi \frac{n}{p} t}$$

Therefore, if we call H the integral function of h, then

$$H = \sum_{n=-\infty}^{+\infty} -\frac{p}{2ni\pi} a_n(h) e^{-2i\pi \frac{n}{p} t}$$

which means that the successive integrations of h will converge to a function corresponding to the term of the fundamental frequency of h, and the amplitude will regress with the integration. Therefore, what was true for the function h will stay valid for its primitive function if p is sufficiently small. If the conditions are satisfied, h can be replaced in the concerned areas by $\bar{h}$ without causing a major deviation from the exact behavior. Consequently, periodic functions may be abstracted by their mean values in the appropriate conditions.

Everything explained here for periodic functions can also be considered for stochastic terms of the functions. The analogy can be completed using the mean value of the stochastic function and by replacing the amplitude in a period with the variance of the random variable. In this case, we rely on the law of large numbers to observe the compensation after many pseudo-periods for stochastic functions. After a high number of time steps, the mean value of the random variable should converge to its theoretical mean, and the limited variance limits the potential qualitative change that could be observed. Therefore, we simplify it as earlier by replacing a stochastic function r, verifying these criteria by its expected value m. Observing an artifact is still possible because the standard deviation does not give us a maximum amplitude but only a probabilistic value of the observed deviation to the expectancy. Nothing prevents a random variable from having an unpredictable peak value way further from the expectancy than the variance made us think. This approximation is less safe than the previous ones, but it still gives a tool to help the system abstraction by transforming non-derivable terms into constant monomials, allowing resolution and derivation, both impossible with purely stochastic terms.

In brief, when faced with a periodic or stochastic term h in an equation, we compute its mean value and compare it to v_q. If $\bar{h}$ is negligible compared to this value, we consider it as 0. In any case, we must then examine the amplitude/variance of the term and its period/sampling period. If both verify the described criteria, their qualitative impact can be simplified to the mean value of the term without considering the variation around it, which will only cause local disturbances.

8.4 . Case Study and Illustration

In the following section, we present an illustration of the developed propositions. As we did not have enough time to automatize the processes for abstracting periodic and stochastic functions, the described examples were tested by manually computing the abstract functions using the rules and criteria mentioned above.

8.4.1 . Windy ball and equation simplification

As in chapter 6, we made our experiments in Python to benefit from the various computation and display libraries and options. We manually compute the abstraction of the functions we presented earlier because of the difficulty of symbolically accessing the information about the function that is necessary to abstract periodic or stochastic behaviors.

To illustrate the interest of our proposition, we present two case studies on a *windy ball* system and a *Van der Pol* oscillator (defined in the equation system 6.1). The first one consists of a ball bouncing on a flat floor and exposed to the effects of an irregular wind.

We use a model where the evolution of the ball follows the ordinary differential system 8.1:

$$\begin{cases} \dot{x} = 10(1 + 0.5sin(10t)) \\ \ddot{y} = -g \end{cases} \tag{8.1}$$

Starting from an initial point $(x_0, y_0) = (0, 10)$ and making a numerical simulation from $t = 0$ to $t = 7$ with a time step of 0.007, using a classical second-order Runge-Kutta integration method, we get the result shown in blue in Figure 8.1.

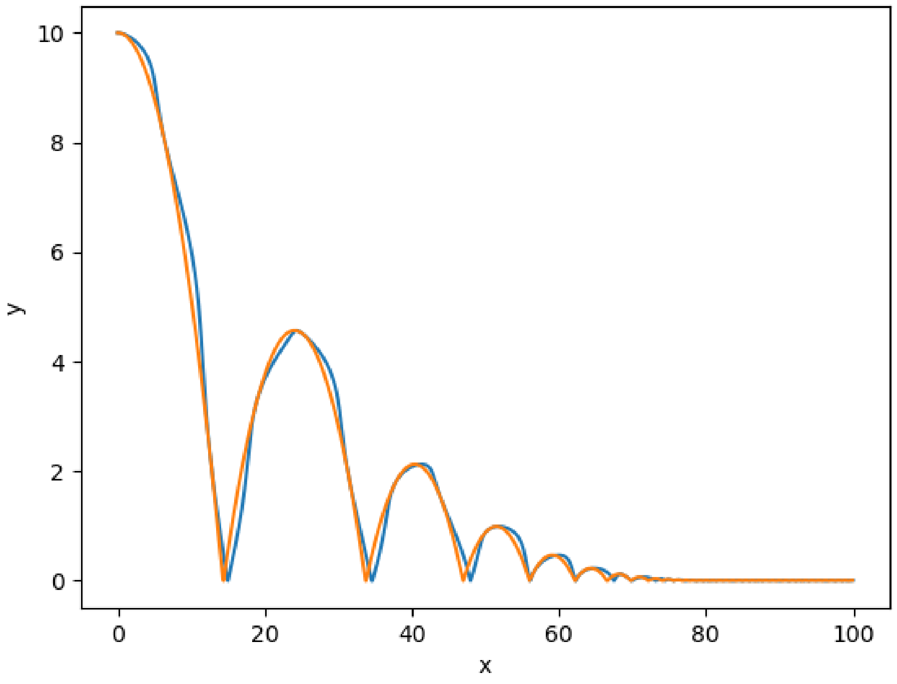

Figure 8.1: Windy ball: suppressing the periodic term (orange) vs. real behavior (blue)

As the term based on the sin function is periodic with mean value $m = 0$ and maximal amplitude 5, we can suppress this term from the differential equation of $\dot{x}$ and compare the new result to the first one as shown in orange in Figure 8.1. This shows that if the position of the impacts is not exact because of the deviation caused by the $\sin$ term in the real system, its periodic nature makes the deviations compensate for each other. The tendencies are well respected, leading to a qualitative reading of the simulation that is very close to reality.

It is possible to modify the system's dynamics to test more of our statements. First, we can replace the sine with a stochastic term to compare the new behavior with its approximation. In Figure 8.2, we can see that with a normal term defined by a Gaussian function $gauss(\mu, \sigma)$ with $\mu = 0.1$ and $\sigma = 1$

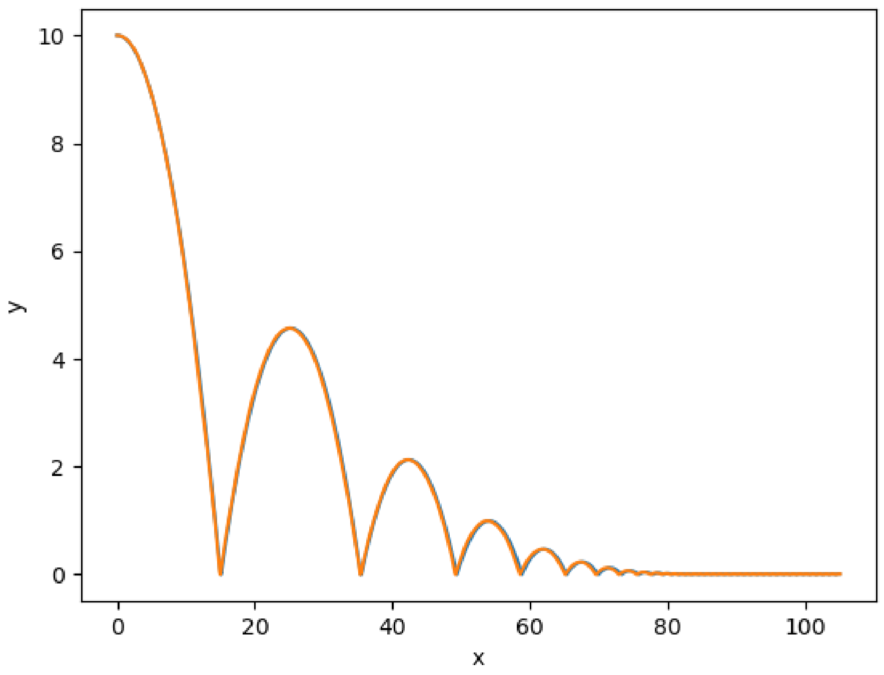

Figure 8.2: Windy ball system: suppressing the stochastic term (orange) vs. real behavior (blue)

and its approximation with $g(t) = \mu$, we get very close results. This shows that it is possible to consider only the value of μ when the exposed criteria are satisfied.

We can also combine the two disturbing factors to see whether our dominance reduction is additive. We show the result in Figure 8.3. Our approximation still efficiently captures the qualitative behavior of a periodic and stochastic differential equation with a simpler *ODE*.

Finally, we make the system more complex and transform it into a hybrid system with different equations for the wind depending on the altitude of the ball. To make this model hybrid, we suppose that the change in the wind effect is discrete and happens at a precise altitude. The equation of $\ddot{y}$ does not change, but the dynamics of $\dot{x}$ becomes:

$$\dot{x} = \begin{cases} 10(1 + 0.5sin(10t)) \text{ if } y > 2 \\ 11 - y^2 \text{ otherwise} \end{cases}$$

The precise integration and its approximation are compared in Figure 8.4. We can observe a little deviation around the third bound, but it disappears for the next ones and does not change the qualitative behavior. It will only change the location of some landmarks.

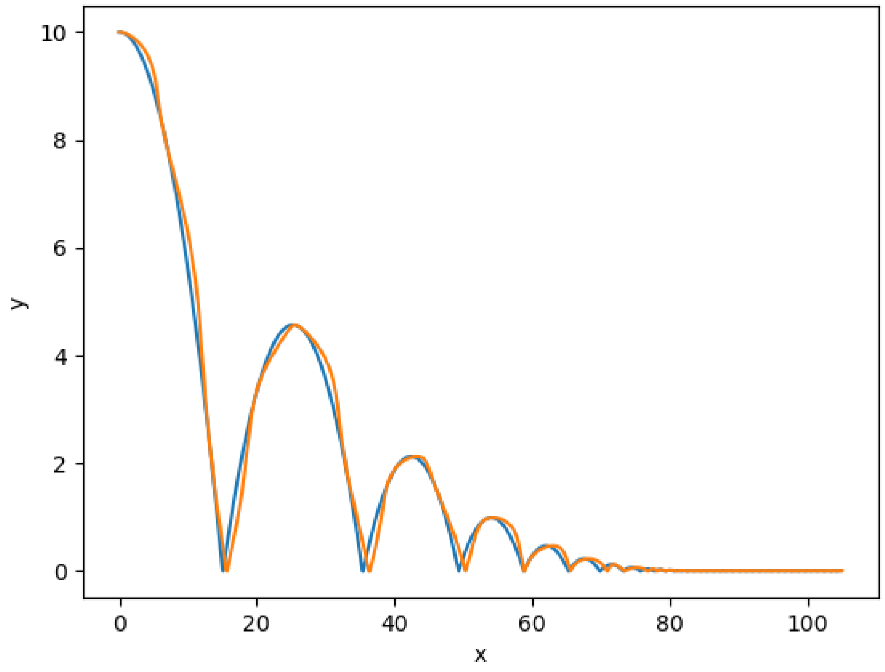

Figure 8.3: Windy ball system: suppressing both periodic and stochastic terms vs. real behavior

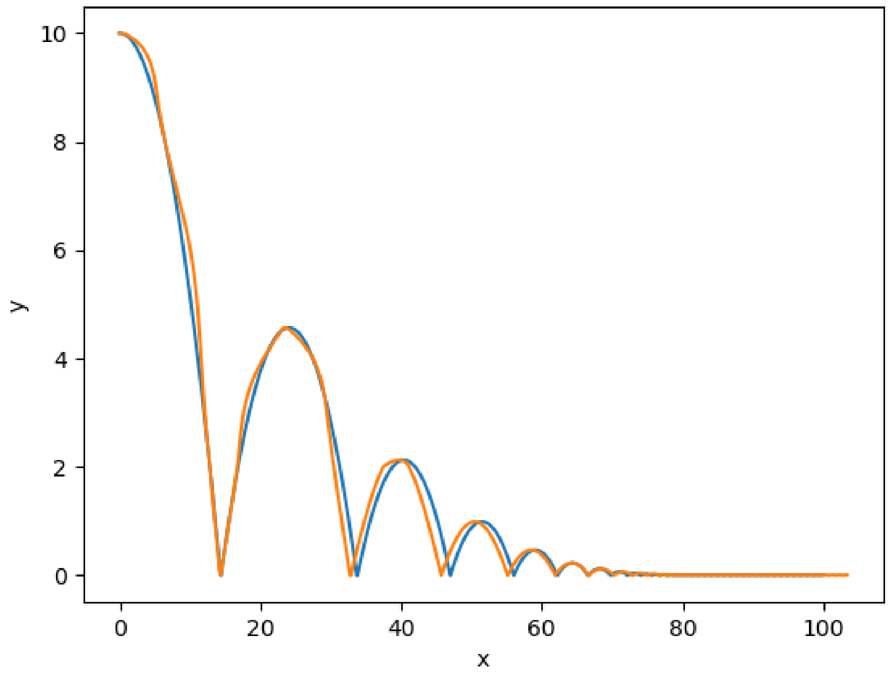

Figure 8.4: Windy ball: hybrid system with wind shear

8.4.2 . Van der pol and qualitative behavior

To highlight the interest of our proposition for qualitative behavior studies, we will apply it to the Van der Pol oscillator, which is a continuous two-dimensional physical system described by the two differential equations in 6.1 for the two variables x and y, with b a positive constant parameter. In this system, the values may oscillate depending on the value of b. As the oscillations happen around small values of x and y, we place the same characteristic value q for both variables at 1 and choose a symmetric 2-logarithmic scale on each of them, on both positive and negative values, which create a negligibility criterion of $\frac{1}{2}$. Therefore, by applying our algorithm to these elements, we can propose a simplified version of the flow, which will then be expressed as:

$$\begin{cases} \dot{x} = \begin{cases} 10(x+y) \text{ if } |x| < 0.5 \\ 10(y - \frac{x^3}{3}) \text{ if } |x| > 4 \\ 10(y + x - \frac{x^3}{3}) \text{ otherwise} \end{cases} \\ \dot{y} = \begin{cases} b \text{ if } |x| < b/2 \text{ and } |y| < b/2 \\ b - x \text{ if } |y| < b/2 \text{ and } |x| < 2b \\ b - \frac{3}{4y} \text{ if } |x| < b/2 \text{ and } |y| < 2b \\ -x - \frac{3y}{4} \text{ otherwise} \end{cases} \end{cases}$$

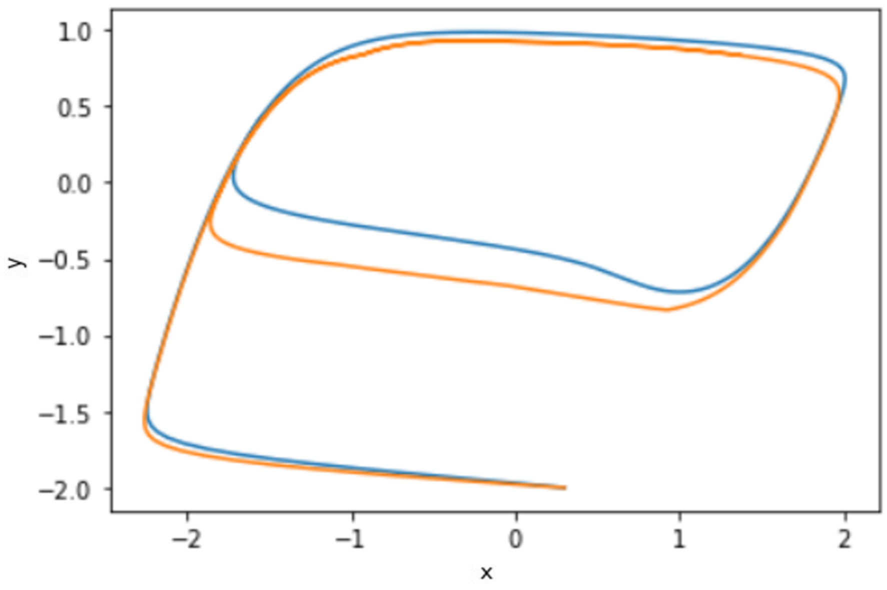

Figure 8.5: Cyclic execution of a Van der Pol system (blue) and its approximation (orange)

We then show the superposition of the curves obtained for a simulation of both systems with the same number of time steps and for two qualitative trajectories as shown in Figure 8.5 and Figure 8.6. The curves of the actual systems are in blue, and the curves of the approximated one are in orange.

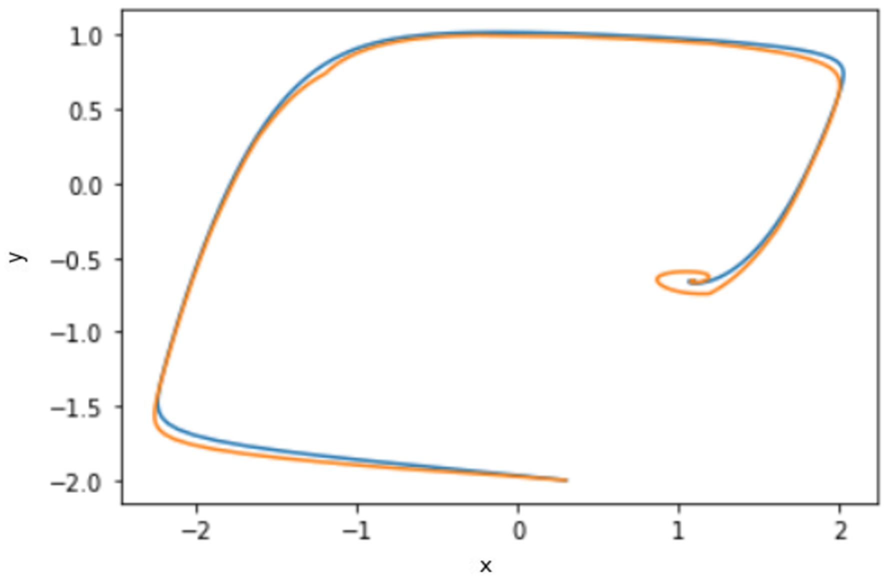

Figure 8.6: Convergent execution of a Van der Pol system (blue) and its approximation (orange)

We can see in both figures that despite an apparent inaccuracy in numerical values, the qualitative behaviors (convergence or limit cycles) are preserved, which is what we were aiming for. Moreover, switching from a 2-logarithmic scale to a 4-logarithmic scale almost completely removes any visible offset between the curves.

8.5 . Limits and Non-Nullifiable Effects

In this paragraph, we will expose the limitations of our contribution, which appear in borderline situations and for very sensitive systems for which it is impossible to apply the dominant term strategy we used before.

The first problem comes from the borderline cases. When the dominated elements are in a border situation, being significantly inferior to the tendency but not below the negligibility frontier or periodic with an amplitude of the same magnitude as the characteristic quantities of the system, the approximation fails. Such terms may cause significant consequences. An example of this situation can be given by replacing the 0.5 factor of the sine function with a 2 in the windy ball case study. The mean of the complete periodic term stays unchanged, but not its amplitude. The maximum distances to the mean are moving from 5 to 20, which is no longer inferior to ten. The comparison of the simulation results in Figure 8.7 illustrates this ambiguity: qualitatively, the

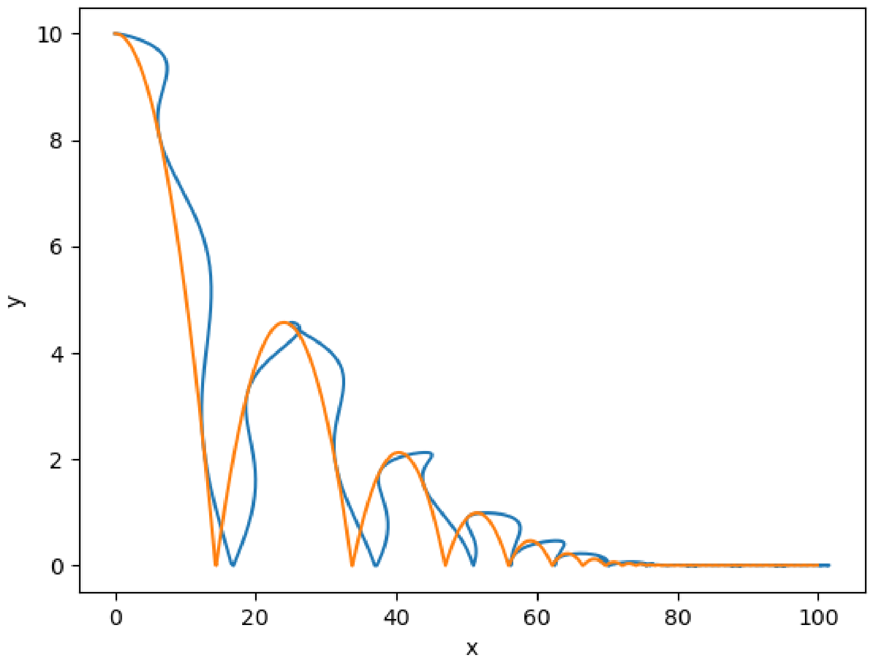

Figure 8.7: Limit of the approximation with $2\sin$.

behavior has the same nature, but the shift of the corresponding landmarks increases and even creates a phase opposition in the approximated behavior. Another example is when the system's parameters are near a behavior-switching condition. For example, in the Van der Pol oscillator presented earlier, when b is close to the switching value between the convergent and divergent behavior, it is possible to observe wrong trajectories.

The second problematic configuration comes from systems where any conditions or equations modification can completely invalidate or qualitatively change the system's behavior. A simple example derived from our previous model is a windy ball with an uneven floor. If we choose a sinusoidal floor, we can emphasize two inherent problems. The first is that it is only possible to apply the nullification to every periodic function with enough knowledge about the internal variables of this function. In our example, the cosine term takes as an argument the x coordinate of the ball, which is itself a non-expressible function. It is impossible to simplify the composed function because we have no clue that $\cos x$ will indeed be periodic as we do not have an explicit expression of x. It is crucial to consider the composed functions, not only the top-level ones. The second problem is the sensitivity to minimal changes in the impact coordinates. A sinusoidal term with a low amplitude can completely change the behavior. This result is illustrated in Figure 8.8.

With a wind of only $0.5\sin(15t)$ added to the dynamics of x in the system, we see that the behaviors of the two models of the same system diverge

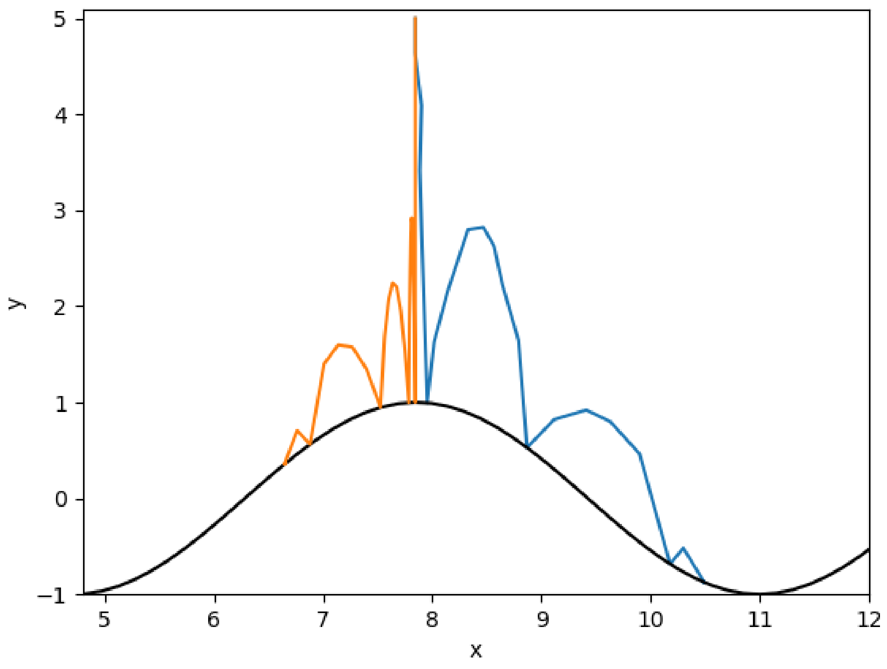

Figure 8.8: Limit of the approximation in the case of chaotic systems.

quickly after the first bounce. This result raises the importance of being aware of the model's sensitivity before applying a dominant simplification.

8.6 . Perspectives

In this section, we present trails of works that we explored to improve the approximation possibilities. However, we did not produce sufficiently solid conclusions to integrate them with our previous contributions. They still have the potential of being investigated and discussed as they were inspired by already proven or employed techniques that can be either applied or improved to fit our reasoning paradigm.

8.6.1 . Polynomial approximation

Once the degrees of the most complex polynomials have been reduced and the periodic and stochastic functions have been simplified to their mean value when possible, the problems come from the terms that enter in none of the previously mentioned categories, such as exponential and logarithmic functions. Considering the formula of orders of growth presented in section 8.2, it appears that $c(f : t \mapsto e^{kt}) = +\infty$ for any constant $k > 0$, meaning that orders of growth are not sufficient to differentiate diverging exponential terms by their relative behavior to ∞. This implies that simplifications cannot be achieved in the same way as with polynomials. Moreover, neither logarithms nor exponential functions give satisfying results when tested with usual polynomial or SMT solvers. Fortunately, it appears that polynomial functions are not just easy to manipulate but also very convenient for function interpolation [142]. This is especially true for a uni-dimensional variable but works on multi-variate functions interpolation has also been proposed [143]. Interpolation is particularly used to give a simple expression of a complex relation between an entry element and its response. As a limited error is often tolerated, interpolation allows precision to be traded for interpretability and computability. Nevertheless, interpolation requires knowing the expression or the values of the initial function on the complete state space. The case of extrapolation is not treated here as polynomials are not particularly adapted for this practice [142]. The first interpolation process generally used in polynomial theory for derivable functions is the Taylor approximation expressed $\forall\, f \in \mathcal{C}^n, \forall\, x_0, h \in \mathbb{K}:$

$$f(x_0 + h) = \sum_{k=0}^{n} \frac{h^k}{k!} f^{(k)}(x_0) + h^n \epsilon(h) \tag{8.2}$$

where ϵ is a function such that $\lim_{h\to 0} \epsilon(h) = 0$ and where $\sum_{k=0}^{n} \frac{h^k}{k!} f^{(k)}(x_0)$ is the Taylor polynomial allowing the approximation.

It has the advantage of presenting a simple and intuitive form, allowing an easy computation for qualitative reasoning. However, the Taylor formula clearly lacks precision when h deviates from 0. This implies that the confidence in the obtained qualitative model will be very low when the state space is too large. Therefore, the best approximation method seems to be the Chebyshev polynomials approximation. The approximation to the N^{th} order of a function g using Chebyshev polynomials can be written as

$$g(x) = \sum_{k=0}^{N} A_k * T_k(x)$$

where T_k is the k^{th} Chebyshev polynomial expressed recursively with

$$\begin{aligned} &\forall\, x \in \mathbb{K},\ T_0(x) = 1, T_1(x) = x \\ &\forall\, k > 1,\ T_k(x) = 2 * x * T_{k-1}(x) - T_{k-2}(x) \\ &\forall\, k < N,\ A_k = \frac{2}{N+1} \sum_{i=0}^{N} g(x_i) * T_k(x_i) \end{aligned}$$

where the x_i are the interpolation points. Many works showed that the Chebyshev approximation polynomial is almost optimum as it is very near the optimal approximation polynomial. As the state space of concrete CPS will very rarely be infinite, it will be possible in a vast majority of the cases to come back to the polynomial case using such an approximation method for general continuous functions.

8.6.2 . Causality reasoning

Finally, one last developed possibility is to unify the different generations of qualitative reasoning on systems to consolidate their respective influence and combine their application areas. Originally, qualitative reasoning was supported by causal ordering [99, 98]. This representation of a system highlights the causality dependence between its variables and components and may integrate a notion of temporality. Mythical causality, for example, uses the temporal ordering between cause and consequence to represent the behavior. Causality applied to qualitative reasoning, as developed by de Kleer and Brown [98], uses QDE, which can be seen as abstractions of *ODE* applied to define the confluences of the system. These qualitative equations associate a sign to a formula composed of simple operations between variables and parameters of the system. This idea of QDE has been used in more recent productions such as [68, 87] to represent as simply as possible some system's properties and to reason on incomplete equations. This can be related to our approach, as the choice to abstract the dynamics of the system is at the core of these contributions. One of the drawbacks of this choice is that when models do not integrate sufficiently usable knowledge, reasoning on it becomes irrelevant because all the achievable conclusions are too simple to present significant interest. A possibility to unify this area of qualitative reasoning to the state space abstraction is to change the formulation of the causality bounds (currently expressed with operators such as $\mathrm{PROP}, \mathrm{CPROP}$ or IPROP) by more standard equations with non valued parameters.

For example, the relation between two variables x and y, represented in the absence of further knowledge as $x \,\mathrm{PROP}\, y$ implies a proportionality relation between x and y. In the formalism exposed in [68], the notion of dynamic dependence is left out on purpose as the authors mainly focus on the sign comparison between the two variables. Consequently, writing $x \,\mathrm{PROP}\, y$ would technically be true in a situation where $x = \frac{1}{y}$, even if the actual relation between x and y exposes an inverse proportionality.

Therefore, developing further this type of formalism to discriminate the static and the dynamic relations could lead to the definition of a more general modeling language that could manage both static and dynamic relations as well as propose different specification precisions for the systems that can be partially or completely known. The anticipated advantage is that these operators could then be translated to symbolic equations or constraints and be used in a model like ours.

To this extent, the operator PROP should be kept for actual dynamic proportionality (the relation that is expressed with $x = cy$ with c a non-valuated positive constant), while the sign similitude between x and y could be expressed using another operator such as SPROP or SSIM. To the same extent, the inverse proportionality could be reserved for two variables such that

$\exists\, c > 0, X = \frac{c}{y}$ which encompass an inverse proportionality relation, while the negative proportionality ($\exists\, c < 0, x = cy$) could be expressed by the operator NPROP. Therefore, using the already defined operators and adapting them to a system that could be expressed with both static and dynamic relations, the formalism could evolve with the given operators:

- PROP defining a dynamic proportionality between two variables i.e. $x\,\mathrm{PROP}\,y \implies \exists\, c > 0,\ x = cy$.
- SPROP defining a symmetric similarity between the signs of the compared variables, i.e., $x\,\mathrm{SPROP}\,y \implies \frac{x}{|x|} = \frac{y}{|y|}$
- NPROP defining a dynamic negative proportionality between two variables i.e. $x\,\mathrm{NPROP}\,y \implies \exists\, c < 0,\ x = cy$.
- IPROP defining a dynamic inverse proportionality between two variables i.e. $x\,\mathrm{IPROP}\,y \implies \exists\, c > 0,\ x = \frac{c}{y}$.
- QPROP defining a dynamic quadratic proportionality between two variables i.e. $x\,\mathrm{QPROP}\,y \implies \exists\, c > 0,\ x = cy^2$.
- MPROP defining a dynamic monomial proportionality between two variables i.e. $x\,\mathrm{MPROP}\,y \implies \exists\, c > 0, n \in \mathbb{N},\ x = cy^n$.
- $\mathrm{P_n}$ defining a polynomial relation of known order between two variables, i.e., $x\,\mathrm{P_n}\,y \implies \exists\, n \in \mathbb{N}, p \in \mathbb{K}_n[X],\ x = p(y)$.
- P defining a polynomial relation of unknown order between two variables, i.e., $x\,\mathrm{P}\,y \implies \exists\, p \in \mathbb{K}[X],\ x = p(y)$.
- R defining a rational relation between two variables i.e. $x\,\mathrm{R}\,y \implies \exists\, f \in \mathbb{K}(X),\ x = f(y)$.
- LE defining the existence of a logarithmic-exponential function between two variables, i.e., $x\,\mathrm{LE}\,y \implies \exists\, f \in \mathcal{E},\ x = f(y)$.
- M defining a monotonous dynamic dependence between variables i.e. $x = M^{+}y$ means that $\exists f$ an increasing function such that $x = f(y)$. On the same logic, M^{-}, M^{++} and M^{--} correspond respectively to the existence of a decreasing function, a strictly increasing and a strictly decreasing function relating x and y.

One can note that some relations are abstractions of others: a monomial is a polynomial, and a proportional relation is a monomial. This hierarchy allows the representation of different qualities of knowledge about the behavior of the system. A system with much knowledge will be expressed using functional dependencies as we used in the state space abstraction process, and a system with very little or unreliable knowledge will exhibit operators that highlight general properties such as P or M.

That would ensure that the literal expression of every relation could be included in a model using polynomial functions to abstract the state space. By enriching the set of allowed operators with, for example, QPROP, $\mathrm{P^nPROP}$, PPROP, REL (meaning respectively the existence of a quadratic, polynomial of the n^{th} order, polynomial of unknown order and unknown but existent relations between variables or components) and by relating each of these operators to a symbolic equation linking the variables to be used in an abstraction process, it is possible to apply the presented system abstraction. The absence of instantiated values for the parameters of the equation stops the possibility of computing transition directions and, therefore, the dynamics of the system but still allows a discretization of the state space and reasoning about the qualitative position on the qualitative states of the system.

9 - RELATED WORKS

9.1 . System State Space Abstraction

In [35], Tiwari introduced qualitative reasoning on hybrid automata using the system's dynamics expressed as polynomial *ODE*s. Any system represented by a hybrid automaton, including such expression of its flow condition, is relevant in this paper as the authors introduce a new automaton structure that they relate to hybrid automata but without further formalization. We further developed this newly introduced automaton structure, formalizing it as a tool to apply qualitative reasoning to systems state spaces. We also highlighted the structural limits of the presented abstraction process as the polynomial *ODE*s have very few chances of being either idempotent or nilpotent. Moreover, we improved its qualitative model formalism by adding the concept of qualitative zones to allow for further reasoning capabilities and applications.

In [144], the authors presented an abstraction process of the system state space using areas of interest and borders that are de facto invariant in order to prove the specificity of the trajectories. The considered areas exhibit borders whose Lie derivatives allow an inward transition, which limits the potential studies. In our contribution, we deal with more general areas and properties, allowing for a more complete study of the system's behavior.

In [26], the authors present a methodology and a formalism that extends Tiwari's, but that is still limited in its reasoning tools as it relies on the study of qualitative states and on the form of the system's equations. In our work, we developed the applicability of this paradigm and new reasoning tools by introducing the concept of qualitative zones and by proposing abstraction methods to deal with the abstraction of the qualitative tendencies rather than with the actual equation if they are not convenient for such a resolution.

The different productions dealing with QSS [132, 133, 134] developed a method to discretize the state space of a system depending on the evolution of its variables. Our works proposed to further push this logic in order to adapt the state space quantization also to the relative position of the system to the different qualitative borders computed with the presented abstraction process in order to avoid unplanned qualitative transitions and keep as much knowledge as possible in a simulation without needing rollback to detect the exact timing of a qualitative event.

9.2 . Behavioral Abstraction

The concept of qualitative simulation, as introduced in [145] and developed in [69, 146], showed interesting results in the representation of the behavior of a given system using sign algebra and variation tables. It allows for general demonstrations that behavior may vary in a certain direction or stay in a certain bound, but it has strong limits when one is interested in more specific properties or more complex invariant conditions that cannot be represented using landmarks. Our contribution, by analyzing more complex inequalities than comparison of a variable with zero and using the discretization process we presented, gives far more concrete knowledge about the possible behavior of the system without needing numerical computation and, therefore, exhibits strong and reliable properties that are particularly valuable for the early studies and design of CPSs.

The qualitative analysis process presented in [68] improves the classic qualitative simulation paradigm with the introduction of causality and symmetric relations between variables that enrich the reasoning models and generate more representative models of systems. However, it still lacks reasoning capabilities to prove behavioral properties that are more complex than the expected ordering of qualitative states in the obtained qualitative behavior. The represented behavior lacks the required precision to support interesting demonstrations and applications as the deduced knowledge is very simple or incomplete.

The diagnosis method proposed in [94] by Mosterman develops an efficient process to represent the intended temporal and causal behavior of a system, allowing fault detection and backpropagation to locate the origin of the faulty trajectory and its cause. The limits stand in the fact that the first representation is mainly used as a reference behavior and cannot be used to support and prove more complex and interesting properties that are not explicitly included in the system's expression. Qualitative abstraction of the system's state space gives the possibility to serve both as a support to create a reference behavior tree from which any deviation may be considered as a faulty behavior but also as a tool to demonstrate critical properties that will particularly make sense will specific use cases and as a solid assistance to design the parameters of the system before any use or application.

9.3 . Design Space Exploration

Some works have studied the different possibilities offered by qualitative reasoning. In [89], the authors focused on the description and the prediction of the behavior of systems at a high level of abstraction, while in [34], Zaatiti developed the diagnosis aspect of qualitative reasoning using more numerical methods. In [147], we proposed a technique to automatize system abstraction and the creation of hybrid automata. All these projects focused on the state space study and showed the advantages and the limits of the different approaches.

Many works proposed various approaches regarding DSE, from the classic search in a finite design space as in [78] to works presenting deep-learning-based strategies as in [138]. Methods also exist to deal with continuous sets using probability density functions as with the Uranie platform [139] to represent the design space based on previous knowledge and more qualitative constraints regarding the expected value of parameters. However, this approach is more appropriate to deal with the choice of the initial value of variables and is, therefore, at the edge between state and design space exploration. Also, many works regarding the solving of temporal logic constraints and its use for optimization have been led (for instance [148]) and may allow significant progress in the resolution of temporal and modal predicates. DSE, in its aspect of design optimization, has been more significantly treated in [149, 150].

10 - CONCLUSION

10.1 . Summary of Results

In this thesis, we were interested in improving the toolbox and the reasoning methods for qualitative modeling of cyber-physical systems using abstraction of the state space based on ordinary differential equations dynamics and rational equations constraints. We based our work on an existing discretization method, which we completed with the concept of qualitative zones. We created a structure named "qualitative automaton" that is adapted to compute the complete qualitative behavior of a system based on its abstraction and on the resolution of the constraints associated with the transitions between the obtained qualitative states. We focused on systems described by polynomial or rational equations as these forms allow for more symbolic computing and resolutions than logarithmic or exponential functions. The computation of these automata required the use of various mathematical methods, such as continuity theorems or the Lie derivative.

We created a prototype tool to automatize the system's state space abstraction and the computation of the behavior tree based on the system's equations. We also proposed a solution to add new constraints in systems to refine the qualitative abstraction to generate the representation that is the most adapted to the considered application. This tool has been tested on different use cases of different complexity and dimensions to challenge its reliability on systems of different natures.

We presented diverse modeling and reasoning paradigms related to CPSs and proposed new solutions to upgrade the possibilities of qualitative models based on our literature review. We also explored different applications of our works on state space abstraction, and we developed some algorithms that fit with our qualitative automata to use them for property proof, trajectory verification, and simulation piloting. These programs can be considered as a supplement to the existing toolbox to study, verify, and understand CPSs using the reasoning opportunities offered by qualitative models.

We briefly studied the possibilities offered by qualitative modeling in the design of CPSs, particularly in choosing the values of the system's parameters. This research direction is still insufficiently developed, but we proposed algorithms that could be associated with qualitative reasoning to allow improvement in the design process of CPSs.

Finally, we studied the possibility of generalizing our work to more complex or less convenient systems with dynamics that do not fit the constraints of having a polynomial or a rational expression. We considered the polynomial interpolation of logarithmic-exponential functions, studied the negligibility of stochastic dynamics terms, and proposed a unification between causal relations and functional dependencies using a new set of causal relations that can be transformed into symbolic functions.

10.2 . Limits and Perspectives

Our work still faces some limits, and many challenges remain to meet the need for more permissive and efficient modeling paradigms and tools. Among them, the most notable are:

- Currently, the main limitations of our tool involve the need to choose and fix the abstraction parameters, whose most adapted value depends both on the system itself and its supposed use case, which is not something intrinsic to the system's expression. Therefore, to fully automatize the abstraction process and remove human intervention in the choice of these parameters, it will be necessary to develop a program able to analyze the system, its environment, its programmed objective, and its associated constraints in order to adapt the modeling parameters and to come up with a qualitative model that would be the most adapted to the specific situation.

- The second problem, and the most important to solve, is the ability to deal with partially instantiated and defined systems. Currently, we must know all the system's parameters to compute the behavior tree, as none of the SMT solvers we tried could solve constraints with both variables and unknown parameters. This is a clear lack of flexibility, as qualitative reasoning is particularly interesting at the early stages of systems development. Therefore, a significant improvement would be allowed by the availability of a solver to address constraints with different levels of unknowns, where the variables would be unknowns of the first level, whose values may be expressed depending on the parameters, which would then be unknowns of a second level. Currently, we are limited to modeling systems with valued parameters to test their conformance and reliability or to create an incomplete model to be used specifically to choose a set of valid parameters. Being able to reason about the behavior of a system with symbolic values for its parameters would be a major improvement in the relevance and use of qualitative reasoning.

- Another important perspective is that we could not automatize the functional abstraction for equations that are not polynomial or rational expressions. We implemented a function that could approximate the most complex polynomials with multivariate terms to more simple expressions by considering the qualitative tendencies as we exposed them, but we did not automatize the mode division based on this new expression of the flow equation. Implementing the hybridization of the considered systems is the next step to creating a complete qualitative modeling tool.

- One of the limits of this contribution is the requirement of having a sufficiently precise expression of the system to provide its abstraction and behavioral study. We are not yet able to solve the situation where one needs to analyze a system about which very little is known. This situation can happen, for example, during a certification process or quality control, when someone who may not have access to a system's full specification or description has to be knowledgeable about its reliability or robustness. In this situation, our process would not be applicable given that no equation could describe the system's dynamics, which is just empirically known to behave in a certain way. To solve this problem, a solution would be to combine our algorithms with solutions of CPS identification as presented in [151]. CPS identification consists of inferring the flow equation of a system based on its numerical trajectory, using methods of pattern matching to deduce the concrete equations defining the system. Combining CPS identification with qualitative reasoning would allow us to study and verify systems that we can only access via their traces and which we cannot be certain of the structure of the flow conditions. It would, therefore, be of great interest to develop the use of qualitative reasoning to further use cases and applications.

- Most of the systems we dealt with in our contributions are deterministic systems or systems exhibiting stochastic elements that can be either neglected or approximated. Unfortunately, many systems do not fit these categories and imply non-negligible stochastic terms or even evolve in non-deterministic or uncertain environments. Such a situation is far more complex to manage as we cannot apply the developed elements if the system may evolve in a completely uncertain direction depending on an environment we cannot control or predict. To this extent, some works such as [152] introduced the concept of Bayesian hybrid automata to deal with this uncertainty and to represent the control of the system depending on the environment. It uses a collaboration of three different automata, respectively, dedicated to the representation of the environment, its estimation by the sensors of the studied CPS, and the control and evolution of this system depending on the returned information from the previous automaton. Applying qualitative reasoning in an uncertain environment could, therefore, be achieved by using qualitative modeling on this set of three automata to abstract their respective state space and evolution as well as their relations and dependencies. This would represent a major challenge as the three automata may have completely different state spaces and, therefore, different abstractions. It would then require a harmonization procedure so that the different abstracted automata can still interact with information that could be shared and understood by all three. More generally, every system that will be subject to a control procedure that implies uncertainty at any moment (be it caused by an uncertain environment, by the low reliability of sensors, or by the necessity of human interventions in the process) will require more advanced research to adapt qualitative reasoning and abstraction process to specific situations that are difficult to include in classic hybrid automata.

- Finally, generalizing our methodology to a larger group of systems is a major perspective that could definitely unify the different qualitative reasoning methods. An ideal approach would be able to handle systems defined by ordinary differential equations or causal relations but also partial differential equations or even more complex structures such as cyber-physical systems of systems (such an approach would imply managing different automata that would interact and influence each other). It would allow scientists to apply qualitative models to far more complex and sophisticated systems like connected cities, air traffic and plane piloting, or energy networks coupled with power plants and distributors.

10.3 . Publications

Conference workshops

- *Qualitative models for the supervision of CPS simulations*, Baptiste Gueuziec, Jean-Pierre Gallois, Frédéric Boulanger, Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 2022, Association for Computing Machinery, page 612–616, [125],
https://doi.org/10.1145/3550356.3561594
- *Qualitative reasoning and cyber-physical systems: abstraction, modeling, and optimized simulation*, Baptiste Gueuziec, Jean-Pierre Gallois, Frédéric Boulanger, MoDeVVa 2023 - 20th workshop on model driven engineering, verification and validation, Västerås [Sweden], Sweden, Oct, 2023, pages 781-790, [147],
https://doi.org/10.1109/MODELS-C59198.2023.00126
- *Qualitative tendencies for hybrid system simulation*, Baptiste Gueuziec, Jean-Pierre Gallois, Frédéric Boulanger, MPM4CPS 2023 - 26th International conference on model driven engineering languages and systems, Västerås, Sweden, Oct, 2023, pages 500-509, [141],
https://doi.org/10.1109/MODELS-C59198.2023.00087

National conference

- *Abstraction qualitative et surveillance de systèmes cyber-physiques*, Baptiste Gueuziec, Jean-Pierre Gallois, Frédéric Boulanger, Modélisation des Systèmes Réactifs, Toulouse, France, Nov, 2023, [153],
https://hal.science/hal-04255640

International conference

- *Qualitative Reasoning and Design Space Exploration*, Baptiste Gueuziec, Jean-Pierre Gallois, Frédéric Boulanger, MODELSWARD 2024, Rome, Italy, Feb, 2024, pages 203-210, [154],
https://doi.org/10.5220/0012417300003645

Journal

- *Qualitative reasoning and cyber-physical systems: abstraction, modeling, and optimized simulation*, Baptiste Gueuziec, Jean-Pierre Gallois, Frédéric Boulanger, Innovations in Systems and Software Engineering, 2024, [120], https://doi.org/10.1007/s11334-024-00567-0

Bibliography

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

[2] E. A. Lee, “The past, present and future of cyber-physical systems: A focus on models,” *Sensors*, vol. 15, no. 3, pp. 4837–4869, 2015.

[3] R. Alur, *Principles of cyber-physical systems*. MIT press, 2015.

[4] K. Berkenkötter, S. Bisanz, U. Hannemann, and J. Peleska, “The hybriduml profile for uml 2.0,” *International Journal on Software Tools for Technology Transfer*, vol. 8, pp. 167–176, 2006.

[5] T. A. Henzinger, “The theory of hybrid automata,” in *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pp. 278–292, IEEE, 1996.

[6] 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*, pp. 209–229, Springer, 1991.

[7] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, “The algorithmic analysis of hybrid systems,” *Theoretical computer science*, vol. 138, no. 1, pp. 3–34, 1995.

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

[9] S. C. Suh, U. J. Tanik, J. N. Carbone, and A. Eroglu, *Applied cyber-physical systems*. Springer, 2014.

[10] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, “Cyber-physical systems: the next computing revolution,” in *Proceedings of the 47th design automation conference*, pp. 731–736, 2010.

[11] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, “What’s decidable about hybrid automata?,” in *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pp. 373–382, 1995.

[12] M. S. Branicky, *Studies in hybrid systems: Modeling, analysis, and control*. PhD thesis, Massachusetts Institute of Technology, 1995.

[13] B. K. Aichernig, H. Brandl, and W. Krenn, "Qualitative action systems," in *International Conference on Formal Engineering Methods*, pp. 206–225, Springer, 2009.

[14] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System dynamics: modeling, simulation, and control of mechatronic systems*. John Wiley & Sons, 2012.

[15] F. E. Cellier and J. Greifeneder, *Continuous system modeling*. Springer Science & Business Media, 2013.

[16] S. Ault and E. Holmgreen, "Dynamics of the brusselator," *Math 715 Projects (Autumn 2002)*, vol. 2, 2003.

[17] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2008.

[18] L. Ye and P. Dague, "Diagnosability analysis of discrete event systems with autonomous components.," in *ECAI*, pp. 105–110, 2010.

[19] S. Gulwani and A. Tiwari, "Constraint-based approach for analysis of hybrid systems," in *International Conference on Computer Aided Verification*, pp. 190–203, Springer, 2008.

[20] O. Bouissou and M. Martel, "Static analysis by abstract interpretation of hybrid systems," *Soumis à Journal of Higher Order and Symbolic Computation*, 2006.

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

[22] N. Lynch, R. Segala, and F. Vaandrager, "Hybrid i/o automata," *Information and computation*, vol. 185, no. 1, pp. 105–157, 2003.

[23] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Hytech: A model checker for hybrid systems," in *Computer Aided Verification: 9th International Conference, CAV'97 Haifa, Israel, June 22–25, 1997 Proceedings 9*, pp. 460–463, Springer, 1997.

[24] S. Batis, *Commande d'une classe de systèmes hybrides par automates hybrides rectangulaires*. PhD thesis, Université de Grenoble, 2013.

[25] I. K. G. Hassapis, "The impact of hybrid automata on system modeling and analysis," 2004.

[26] H. Zaatiti, *Modélisation et simulation qualitative de systemes hybrides*. PhD thesis, Université Paris-Saclay (ComUE), 2018.

[27] P. S. Duggirala and S. Mitra, "Lyapunov abstractions for inevitability of hybrid systems," in *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pp. 115–124, 2012.

[28] M. Kloetzer and C. Belta, "Reachability analysis of multi-affine systems," in *International Workshop on Hybrid Systems: Computation and Control*, pp. 348–362, Springer, 2006.

[29] M. Benedikt, T. Duff, A. Sharad, and J. Worrell, "Polynomial automata: Zeroness and applications," in *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–12, IEEE, 2017.

[30] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

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

[32] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine, "Kronos: A model-checking tool for real-time systems," in *Computer Aided Verification: 10th International Conference, CAV'98 Vancouver, BC, Canada, June 28–July 2, 1998 Proceedings 10*, pp. 546–550, Springer, 1998.

[33] P. Bouyer, M. Colange, and N. Markey, "Symbolic optimal reachability in weighted timed automata," in *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I 28*, pp. 513–530, Springer, 2016.

[34] 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*, pp. 279–318, Springer, 2018.

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

[36] R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.

[37] A. Quarteroni and A. Valli, *Numerical approximation of partial differential equations*, vol. 23. Springer Science & Business Media, 2008.

[38] W. Taha, A. Duracz, Y. Zeng, K. Atkinson, F. A. Bartha, P. Brauner, J. Duracz, F. Xu, R. Cartwright, M. Konečný, *et al.*, "Acumen: An open-source testbed for cyber-physical systems research," in *Internet of Things. IoT*

Infrastructures: Second International Summit, IoT 360° 2015, Rome, Italy, October 27–29, 2015, Revised Selected Papers, Part I, pp. 118–130, Springer, 2016.

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

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

[41] X. Chen, E. Abraham, and S. Sankaranarayanan, "Taylor model flowpipe construction for non-linear hybrid systems," in *2012 IEEE 33rd Real-Time Systems Symposium*, pp. 183–192, IEEE, 2012.

[42] A. Chutinan and B. H. Krogh, "Computational techniques for hybrid system verification," *IEEE transactions on automatic control*, vol. 48, no. 1, pp. 64–75, 2003.

[43] L. H. De Figueiredo and J. Stolfi, "Affine arithmetic: concepts and applications," *Numerical Algorithms*, vol. 37, pp. 147–158, 2004.

[44] S. M. Watt, "Making computer algebra more symbolic," in *Proc. Transgressive Computing*, pp. 43–49, 2006.

[45] R. Liska, M. Y. Shashkov, and A. V. Solovjov, "Support-operators method for pde discretization: symbolic algorithms and realization," *Mathematics and Computers in Simulation*, vol. 35, no. 2, pp. 173–183, 1993.

[46] R. Baldoni, E. Coppa, D. C. D'elia, C. Demetrescu, and I. Finocchi, "A survey of symbolic execution techniques," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–39, 2018.

[47] E. Ábrahám, "Building bridges between symbolic computation and satisfiability checking," in *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, pp. 1–6, 2015.

[48] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, *et al.*, "Sympy: symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, 2017.

[49] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, Springer, 2008.

[50] J. C. King, "Symbolic execution and program testing," *Communications of the ACM*, vol. 19, no. 7, pp. 385–394, 1976.

[51] I. Tiraboschi, T. Rezk, and X. Rival, "Sound symbolic execution via abstract interpretation and its application to security," in *International Conference on Verification, Model Checking, and Abstract Interpretation*, pp. 267–295, Springer, 2023.

[52] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli, "cvc4," in *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23*, pp. 171–177, Springer, 2011.

[53] R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani, "The mathsat 4 smt solver: Tool paper," in *Computer Aided Verification: 20th International Conference, CAV 2008 Princeton, NJ, USA, July 7-14, 2008 Proceedings 20*, pp. 299–303, Springer, 2008.

[54] R. Bruttomesso, E. Pek, N. Sharygina, and A. Tsitovich, "The opensmt solver," in *Tools and Algorithms for the Construction and Analysis of Systems: 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings 16*, pp. 150–153, Springer, 2010.

[55] A. Platzer and Y. K. Tan, "How to prove "all" differential equation properties," Tech. Rep. CMU-CS-17-117, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 2017. Extended version at arXiv:1802.01226.pdf.

[56] J.-D. Quesel, S. Mitsch, S. Loos, N. Aréchiga, and A. Platzer, "How to model and prove hybrid systems with keymaera: a tutorial on safety," *International Journal on Software Tools for Technology Transfer*, vol. 18, no. 1, pp. 67–91, 2016.

[57] A. Platzer and J.-D. Quesel, "Keymaera: A hybrid theorem prover for hybrid systems (system description)," in *International Joint Conference on Automated Reasoning*, pp. 171–178, Springer, 2008.

[58] A. Platzer, "Differential dynamic logic for hybrid systems," *Journal of Automated Reasoning*, vol. 41, no. 2, pp. 143–189, 2008.

[59] A. Platzer, *Logical Foundations of Cyber-Physical Systems*. Cham: Springer, 2018.

[60] 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.

[61] M. Sap, V. Shwartz, A. Bosselut, Y. Choi, and D. Roth, “Introductory tutorial: Commonsense reasoning for natural language processing,” *Association for Computational Linguistics (ACL 2020): Tutorial Abstracts*, vol. 27, 2020.

[62] I. Apperly, *Mindreaders: the cognitive basis of" theory of mind"*. Psychology Press, 2010.

[63] B. Smith and R. Casati, “La physique naïve: un essai d’ontologie,” *Intellectica*, vol. 17, no. 2, 1993.

[64] M. Mitchell, *Artificial intelligence: A guide for thinking humans*. Penguin UK, 2019.

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

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

[67] Y. Iwasaki and H. A. Simon, “Causality and model abstraction,” *Artificial intelligence*, vol. 67, no. 1, pp. 143–194, 1994.

[68] S. 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.

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

[70] M. Mitchell and D. R. Hofstadter, “The emergence of understanding in a computer model of concepts and analogy-making,” *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 322–334, 1990.

[71] M. Mitchell, “Abstraction and analogy-making in artificial intelligence,” *Annals of the New York Academy of Sciences*, vol. 1505, no. 1, pp. 79–101, 2021.

[72] P. Derler, E. A. Lee, and A. S. Vincentelli, “Modeling cyber–physical systems,” *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13–28, 2011.

[73] W. Yang, C. Xu, M. Pan, X. Ma, and J. Lu, “Improving verification accuracy of cps by modeling and calibrating interaction uncertainty,” *ACM Transactions on Internet Technology (TOIT)*, vol. 18, no. 2, pp. 1–37, 2018.

[74] E. A. Lee, “Cyber physical systems: Design challenges,” in *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*, pp. 363–369, IEEE, 2008.

[75] I. Graja, S. Kallel, N. Guermouche, S. Cheikhrouhou, and A. Hadj Kacem, “A comprehensive survey on modeling of cyber-physical systems,” *Concurrency and Computation: Practice and Experience*, vol. 32, no. 15, p. e4850, 2020.

[76] M. Aiguier, F. Bretaudeau, and D. Krob, *Complex Systems Design & Management: Proceedings of the First International Conference on Complex Systems Design & Management CSDM 2010*. Springer Science & Business Media, 2010.

[77] B. C. Schafer and Z. Wang, “High-level synthesis design space exploration: Past, present, and future,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2628–2639, 2019.

[78] Z. Lattmann, A. Pop, J. De Kleer, P. Fritzson, B. Janssen, S. Neema, T. Bapty, X. Koutsoukos, M. Klenk, D. Bobrow, *et al.*, “Verification and design exploration through meta tool integration with openmodelica,” in *Proceedings of the 10th International Modelica Conference*, pp. 353–362, 2014.

[79] E. Kang, E. Jackson, and W. Schulte, “An approach for effective design space exploration,” in *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems: 16th Monterey Workshop 2010, Redmond, WA, USA, March 31-April 2, 2010, Revised Selected Papers 16*, pp. 33–54, Springer, 2011.

[80] J. P. Hespanha, “Polynomial stochastic hybrid systems,” in *International Workshop on Hybrid Systems: Computation and Control*, pp. 322–338, Springer, 2005.

[81] J. Hu, J. Lygeros, and S. Sastry, “Towards a theory of stochastic hybrid systems,” in *International Workshop on Hybrid Systems: Computation and Control*, pp. 160–173, Springer, 2000.

[82] A. R. Teel, A. Subbaraman, and A. Sferlazza, “Stability analysis for stochastic hybrid systems: A survey,” *Automatica*, vol. 50, no. 10, pp. 2435–2456, 2014.

[83] M. Sugeno and T. Yasukawa, “A fuzzy-logic-based approach to qualitative modeling,” *IEEE Transactions on fuzzy systems*, vol. 1, no. 1, p. 7, 1993.

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

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

[86] P. J. Hayes, "The naive physics manifesto," *Expert systems in the micro-electronic age*, 1979.

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

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

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

[90] J. De Kleer and J. S. Brown, "A qualitative physics based on confluences," *Artificial intelligence*, vol. 24, no. 1-3, pp. 7–83, 1984.

[91] 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.

[92] B. K. Aichernig, H. Brandl, and F. Wotawa, "Conformance testing of hybrid systems with qualitative reasoning models," *Electronic Notes in Theoretical Computer Science*, vol. 253, no. 2, pp. 53–69, 2009.

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

[94] 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.

[95] H. Brandl, "Testing of hybrid systems using qualitative models," in *Proceedings of Formal Methods 2009 Doctoral Symposium*, p. 46, 2009.

[96] F. Wotavya, "Generating test-cases from qualitative knowledge–preliminary report," 2007.

[97] B. Bredeweg, A. Bouwer, J. Jellema, D. Bertels, F. F. Linnebank, and J. Liem, "Garp3: A new workbench for qualitative reasoning and modelling," in *Proceedings of the 4th international conference on Knowledge capture*, pp. 183–184, 2007.

[98] J. De Kleer and J. S. Brown, "Theories of causal ordering," *Artificial intelligence*, vol. 29, no. 1, pp. 33–61, 1986.

[99] Y. Iwasaki and H. A. Simon, "Causality in device behavior," *Artificial intelligence*, vol. 29, no. 1, pp. 3–32, 1986.

[100] B. Smith and R. Casati, "Naive physics," *Philosophical psychology*, vol. 7, no. 2, pp. 227–247, 1994.

[101] A. A. DiSessa, "Toward an epistemology of physics," *Cognition and instruction*, vol. 10, no. 2-3, pp. 105–225, 1993.

[102] M. Reiner, J. D. Slotta, M. T. Chi, and L. B. Resnick, "Naive physics reasoning: A commitment to substance-based conceptions," *Cognition and instruction*, vol. 18, no. 1, pp. 1–34, 2000.

[103] S. Gao, S. Kong, and E. M. Clarke, "Satisfiability modulo odes," in *2013 Formal Methods in Computer-Aided Design*, pp. 105–112, IEEE, 2013.

[104] J.-P. Gallois and J.-Y. Pierron, "Qualitative simulation and validation of complex hybrid systems," in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[105] T. Hickey, Q. Ju, and M. H. Van Emden, "Interval arithmetic: From principles to implementation," *Journal of the ACM (JACM)*, vol. 48, no. 5, pp. 1038–1068, 2001.

[106] B. Kuipers and D. Berleant, "Using incomplete quantitative knowledge in qualitative reasoning.," in *AAAI*, vol. 88, pp. 324–329, Saint-Paul, MN, 1988.

[107] D. Berleant and B. Kuipers, "Qualitative-numeric simulation with q3," *Recent advances in qualitative physics*, pp. 3–16, 1992.

[108] H. Kay, "Sqsim: a simulator for imprecise ode models," *Computers & chemical engineering*, vol. 23, no. 1, pp. 27–46, 1998.

[109] Q. Shen and R. Leitch, "Fuzzy qualitative simulation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 4, pp. 1038–1061, 1993.

[110] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 21, no. 4, pp. 83–93, 1988.

[111] M. L. Mavrovouniotis and G. Stephanopoulos, "Formal order-of-magnitude reasoning in process engineering," *Computers & Chemical Engineering*, vol. 12, no. 9-10, pp. 867–880, 1988.

[112] N. Piera and L. Travé, "About qualitative equality: Axioms and properties," in *9th International Workshop on Expert System and their Applications*, Avignon, 1989.

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

[114] N. Piera, M. Sanchez, and L. Travé-Massuyès, "Qualitative operators for order-of-magnitude calculers: Robustness and precision," in *Thirteenth IMACS World Congress on Computation and Applied Mathematics*, vol. 22, 1991.

[115] P. Dague, "Numeric reasoning with relative orders of magnitude.," in *AAAI*, pp. 541–547, 1993.

[116] É. Borel and A. Denjoy, *Leçons sur la théorie de la croissance*, vol. 12. Gauthier-Villars, 1910.

[117] G. H. Hardy, "Properties of logarithmico-exponential functions," *Proceedings of the London Mathematical Society*, vol. 2, no. 1, pp. 54–90, 1912.

[118] A. Missier and L. Trave-Massuyes, "Temporal information in qualitative simulation," in *Proceedings The Second Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, pp. 298–299, IEEE Computer Society, 1991.

[119] A. D'Innocenzo, A. A. Julius, M. D. Di Benedetto, and G. J. Pappas, "Approximate timed abstractions of hybrid automata," in *2007 46th IEEE Conference on Decision and Control*, pp. 4045–4050, IEEE, 2007.

[120] B. Gueuziec, J.-P. Gallois, and F. Boulanger, "Qualitative reasoning and cyber-physical systems: abstraction, modeling, and optimized simulation," *Innovations in Systems and Software Engineering*, July 2024.

[121] Y. Gao, N. Moreira, R. Reis, and S. Yu, "A survey on operational state complexity," *arXiv preprint arXiv:1509.03254*, 2015.

[122] O. Sokolsky and H. S. Hong, "Qualitative modeling of hybrid systems," in *Proc. of the Montreal Workshop*, Citeseer, 2001.

[123] R. R. Rosenberg and D. C. Karnopp, *Introduction to physical system dynamics*. McGraw-Hill, Inc., 1983.

[124] K. Yano, *The theory of Lie derivatives and its applications*. New York: Courier Dover Publications, 2020.

[125] B. Gueuziec, J.-P. Gallois, and F. Boulanger, "Qualitative models for the supervision of cps simulations," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pp. 612–616, 2022.

[126] O. Zienkiewicz and Y. Xie, “A simple error estimator and adaptive time stepping procedure for dynamic analysis,” *Earthquake engineering & structural dynamics*, vol. 20, no. 9, pp. 871–887, 1991.

[127] C. K. WAI, L. K. CHOON, and R. IDRIS, “Adaptive time-stepping for runge-kutta methods for ordinary differential equations,” *Universiti Malaysia Terengganu Journal of Undergraduate Research*, vol. 3, no. 1, pp. 25–36, 2021.

[128] A. F. Bastani and S. M. Hosseini, “A new adaptive runge–kutta method for stochastic differential equations,” *Journal of computational and applied mathematics*, vol. 206, no. 2, pp. 631–644, 2007.

[129] H. Lamba, J. C. Mattingly, and A. M. Stuart, “An adaptive euler–maruyama scheme for sdes: convergence and stability,” *IMA journal of numerical analysis*, vol. 27, no. 3, pp. 479–506, 2007.

[130] K. Nguyen, Q. Tran, L. Manin, S. Baguet, and M. Andrianoely, “Un schéma d’intégration temporelle pour la réponse transitoire de systèmes mécaniques avec butées de contact,” 2017.

[131] B. Stout, “Méthodes numériques de résolution d’équations différentielles,” *Marseille, France: Universite de Provence*, 2007.

[132] 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.

[133] 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.

[134] 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*, pp. 657–667, 2011.

[135] J. Zhou, C. Wen, W. Wang, and F. Yang, “Adaptive backstepping control of nonlinear uncertain systems with quantized states,” *IEEE Transactions on Automatic Control*, vol. 64, no. 11, pp. 4756–4763, 2019.

[136] H. Han and R. G. Sanfelice, “Linear temporal logic for hybrid dynamical systems: Characterizations and sufficient conditions,” *Nonlinear Analysis: Hybrid Systems*, vol. 36, p. 100865, 2020.

[137] M. Palesi and T. Givargis, “Multi-objective design space exploration using genetic algorithms,” in *Proceedings of the tenth international symposium on Hardware/software codesign*, pp. 67–72, 2002.

[138] M. Motamedi, P. Gysel, V. Akella, and S. Ghiasi, "Design space exploration of fpga-based deep convolutional neural networks," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 575–580, IEEE, 2016.

[139] J.-B. Blanchard, G. Damblin, J.-M. Martinez, G. Arnaud, and F. Gaudier, "The uranie platform: an open-source software for optimisation, meta-modelling and uncertainty analysis," *arXiv preprint arXiv:1803.10656*, 2018.

[140] E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Informatica*, vol. 43, pp. 451–476, 2007.

[141] B. Gueuziec, F. Boulanger, and J.-P. Gallois, "Qualitative tendencies for hybrid system simulation," in *MPM4CPS 2023-Multi-Paradigm Modelling for Cyber-Physical Systems-26th International conference on model driven engineering languages and systems*, 2023.

[142] G. K. Smyth, "Polynomial approximation," *Encyclopedia of Biostatistics*, vol. 13, 1998.

[143] M. Gasca and T. Sauer, "Polynomial interpolation in several variables," *Advances in Computational Mathematics*, vol. 12, pp. 377–410, 2000.

[144] C. Sloth and R. Wisniewski, "Complete abstractions of dynamical systems by timed automata," *Nonlinear Analysis: Hybrid Systems*, vol. 7, no. 1, pp. 80–100, 2013.

[145] K. D. Forbus, *Introducing actions into qualitative simulation*. No. 1451-1465, Qualitative Reasoning Group, Department of Computer Science, University of Illinois, 1988.

[146] B. Kuipers, "The limits of qualitative simulation.," in *IJCAI*, vol. 9, Los Angeles, 1985.

[147] B. Gueuziec, J.-P. Gallois, and F. Boulanger, "Qualitative reasoning and cyber-physical systems: abstraction, modeling, and optimized simulation," in *MoDeVVa 2023-20th workshop on model driven engineering, verification and validation*, 2023.

[148] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5319–5325, IEEE, 2014.

[149] M. Fuchs and A. Neumaier, "Discrete search in design optimization," in *Complex Systems Design & Management: Proceedings of the First International Conference on Complex System Design & Management CSDM 2010*, pp. 113–122, Springer, 2010.

[150] G. G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 4255, pp. 415–426, 2006.

[151] Y. Monier, G. Faraut, B. Denis, and N. Anwer, "Inferring moore machine for adaptive online hybrid automaton identification," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 8641–8647, 2023.

[152] P. Kröger and M. Fränzle, "Bayesian hybrid automata: A formal model of justified belief in interacting hybrid systems subject to imprecise observation," 2022.

[153] B. Gueuziec, J.-P. Gallois, and F. Boulanger, "Abstraction qualitative et surveillance de systèmes cyber-physiques," in *Modélisation des Systèmes Réactifs*, 2023.

[154] B. Gueuziec, J.-P. Gallois, and F. Boulanger, "Qualitative reasoning and design space exploration," in *MODELSWARD 2024*, 2024.