

# Qualitative Reasoning and Design Space Exploration

Baptiste Gueuziec<sup>1</sup><sup>a</sup>, Jean-Pierre Gallois<sup>1</sup><sup>b</sup> and Frédéric Boulanger<sup>2</sup><sup>c</sup>

<sup>1</sup>Université Paris-Saclay, CEA List, F-91120 Palaiseau, France

<sup>2</sup>Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire Méthodes Formelles, F-91190, Gif-sur-Yvette, France

**Keywords:** Design Space Exploration, Qualitative Reasoning, Constraint Solving.


**Abstract:** The design of complex systems is a challenging task, combining optimization and testing techniques. Design space exploration allows the designer to optimize the parameters of the system, but is usually very costly and induces many inherent difficulties that require specific computation techniques to be solved. Qualitative reasoning was first introduced to describe system structures and causality, and developed to study the behavior of the system and discretize its state space into qualitative states. It is now used in diagnosis and verification to reduce computation time and precision while still preserving major properties of the behavior. This article presents how we think that qualitative reasoning can be applied to design space exploration in addition to state space discretization, i.e. how it may help in the choice of parameters for a system, by reducing the computational cost of this exploration.


## 1 INTRODUCTION


Qualitative reasoning is the area of computer science and engineering that focuses on the representation and reasoning on models with very little and imprecise information (Forbus, 2014). Our previous works on this topic were applied to the study of hybrid systems and allowed us to study the system's state space based on its dynamics, constraints, and discrete controller. We obtained an entirely qualitative model of the behavior of a system, discretizing the state space into qualitative states that represent phases of the system's behavior. This discretization allows us to abstract both the state of the system and its behavior. The computation of the qualitative behavior generates path conditions requiring to be solved by symbolic solvers such as *Sympy* and *Z3*. In the instantiated systems, we obtained very satisfying results to study and anticipate the possible trajectories of the system. However, when we try to generalize our approach to the study of the design space, we meet the limits of these tools when computing the transitions. The design space of a system is the space of the possible values of its constant parameters, and its exploration requires more generality and abstraction than the exploration of the state space. Very few computational tools allow

for the symbolic resolution of multivariate inequalities with variables of multiple orders. The difficulty comes from the fact that SMT solvers only find one solution that respects the given constraints, while the exploration of the design space would require finding all the values of the parameters for which there exists a solution in the variable space that satisfies the constraint. For a system with parameters  $P$  and variables  $X$ , exploring the state space amounts to seeking a valuation  $(A, x)$  of  $(P, X)$  satisfying a predicate  $R(A, x)$ . When exploring the design space, we instead look for  $V_P \subset \mathbf{D}_P$  such that  $\forall A \in V_P$ , then  $\exists x \in \mathbf{D}_X$  verifying  $R(A, x)$  where  $\mathbf{D}_P$  and  $\mathbf{D}_X$  are the domains of the parameter  $P$  and the variable  $X$ . This logic problem cannot be solved generally by simply using classic SMT solvers.

Design Space Exploration (DSE) is a critical domain of the conception of complex systems. It consists in choosing, testing, and comparing different configurations of values for the parameters of the system before its construction to verify some property or optimize some utility function (Aiguier et al., 2010). High-level DSE can be considered as multi-objective optimization (Schafer and Wang, 2019) as we may want to optimize a set of conflicting constraints and utility functions. The solutions of the parameters are often searched in a finite space of authorized values (Lattmann et al., 2014), and the arising problem consists both in the explosion of the number of possible

<sup>a</sup> <https://orcid.org/0000-0002-4596-9769>

<sup>b</sup> <https://orcid.org/0000-0001-6982-6247>

<sup>c</sup> <https://orcid.org/0000-0003-3185-2807>

solutions when taking into account all the parameters and in the critical time needed to test the validity of all of these solutions. To this extent, an effective DSE requires three elements: a representation, an analysis, and an exploration method (Kang et al., 2011). The representation of the system must be designed to optimize the research, the test, and the comparison of solutions. The analysis requires a metric, a utility function, or a set of logic predicates to evaluate and compare the different valuations of the parameters of the system. It also requires the choice of a verification method. For this verification task, various methodologies are employed, with different levels of reliability, depending on the complexity of the system and the number of solutions to test. These methods mainly consist of simulating the system to verify a property, measuring an output value, or using some verification process to check the validity of a specific property. Finally, the exploration method is necessary to choose the valuations to test and will strongly depend on the nature of the design space. For example, some contributions have studied the possibility of searching the design space using machine learning methods (Blanchard et al., 2018; Kim et al., 2018; Mudgal et al., 2018), others use brute force search in discrete sets, and others use gradient descent methods in the case of continuous spaces (Blanchard et al., 2018).

## 2 CONTEXT AND INITIAL PROBLEM

### 2.1 Hybrid Systems and Qualitative Modeling

We place ourselves in the context of hybrid cyber-physical systems, represented as:

- $Q$  the current mode of the system (discrete),
- $X$  a set of  $n$  continuous variables,
- $I$  a function mapping each mode to its invariant conditions,
- $F$  a function mapping each mode to its flow conditions,
- $T$  a set of discrete transitions, each described by their starting mode, their condition, their target mode, and their reset function
- $P$  a set of  $m$  parameters

$Q$  takes values on  $\mathbf{D}_Q$ , the finite set of modes of the system,  $X$  is valuated on  $\mathbb{K}^n$  with  $\mathbb{K} = \mathbb{R}$  in the general case, and  $P$  takes values on  $\mathbf{D}_P$  a set of dimension  $m$ .  $\mathbf{D}_P$  can be either continuous or discrete. We make the realistic hypothesis that  $\mathbf{D}_P$  is at least bounded to fa-

cilitate its exploration. We consider that the flow conditions are represented as ordinary differential equations (ODE), i.e.,  $\dot{X} = f(X, t)$  with  $t$  the time parameter. For example, an autonomous car running on a straight road on the axis  $x$  can be represented as:

$$\mathbf{D}_Q = \{Stop, Accelerate, StableSpeed, Decelerate, Reverse\},$$

$$X = (x, \dot{x}) \text{ with } x \text{ the coordinate of the car on the variation axis and } \dot{x} \text{ its speed on this axis,}$$

$$\mathbf{X} = \mathbb{K}^2 = \mathbb{R}^2,$$

$$I = \{Stop : X = 0 ; Accelerate : \dot{x} > 0 ; StableSpeed : \dot{x} > 0 ; Decelerate : \dot{x} > 0 ; Reverse : \dot{x} < 0\},$$

$$F = \{Stop : \dot{X} = 0 ;$$

$$Accelerate : \dot{X} = X \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + (0 \ a) ;$$

$$StableSpeed : \dot{X} = X \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + (0 \ 0) ;$$

$$Decelerate : \dot{X} = X \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + (0 \ b) ;$$

$$Reverse : \dot{X} = X \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + (0 \ 0)\}$$

with  $a > 0$ , and  $b < 0$

$$T = \{(Stop, \dot{x} > \varepsilon, Accelerate, X \mapsto X), (Accelerate, \dot{x} = v_{max}, StableSpeed, X \mapsto X), (StableSpeed, x > x_{max} - \gamma, Decelerate, X \mapsto X), (Decelerate, \dot{x} < \varepsilon, Stop, X \mapsto (x, 0)), (Stop, \dot{x} < -\varepsilon, Reverse, X \mapsto X), (Reverse, \dot{x} > -\varepsilon, Stop, X \mapsto (x, 0))\}$$

with  $\varepsilon > 0$  a negligibility criterion under which a value is considered as null,  $\gamma$  a security distance from the objective point  $x_{max}$  and  $v_{max}$  the maximal speed allowed.

$$P = \{a, b, c, \varepsilon, \gamma, x_{max}, v_{max}\}$$

Qualitative modeling consists in the creation of a model adapted to support qualitative reasoning (Sugeno and Yasukawa, 1993). As this reasoning paradigm has the ambition to avoid numerical elements, the stake of qualitative modeling is to design models of Cyber-Physical-Systems that integrate enough information from the system to perform efficient reasoning without using numerical knowledge or computation. We call *qualitative model* the model resulting from this design. A qualitative model must not consider variables as numerical valuations but as sets of values. For example, the autonomous car represented in subsection 2.1 can be qualitatively represented by imposing:

$$\mathbf{X} = \{-, 0, +\};$$

$$F = \{Stop : \dot{X} = 0 ;$$

$$Accelerate : \dot{X} = (++) ;$$

$$StableSpeed : \dot{X} = (+0) ;$$

$$\begin{aligned} \text{Decelerate} : \dot{X} &= (+-); \\ \text{Reverse} : \dot{X} &= (-0) \end{aligned}$$

In our previous works (Gueuziec et al., 2023), we studied the suitability of qualitative reasoning (Kuipers, 1986; Tiwari and Khanna, 2002) to represent, study, and refine the state space of hybrid systems. We used polynomial solving to create a qualitative map of this state space and SMT solvers to determine all the possible transitions between the obtained qualitative states. This computation allowed us to generate a complete behavioral tree of the system between its modes and qualitative states. However, the limit of this approach is that it requires all the parameters of  $P$  to be known. Considering that one of the main causes of interest in qualitative reasoning is its high level of abstraction, its use in a context where all the values of  $P$  are not entirely defined seems appropriate. However, since the computation of the transitions in the state space of the system requires solving symbolic inequalities with a SMT solver (we use Z3 in our model), the presence of symbolic constants in the equations of the system creates significant problems. These symbolic constants should not be treated on the same logic level, and using a simple SMT solver as previously done is not sufficient anymore because we are now searching for a subset  $V_P \subset \mathbf{D}_P$  of parameter values such that for any set  $p$  of parameters in  $V_P$ , there exists  $x \in \mathbb{K}^n$  that satisfies a predicate  $R(p, x)$ . For example, if we consider a Van der Pol oscillator system whose dynamics is given by the equation system :

$$\begin{cases} \dot{x} = 10(y + x - \frac{x^3}{3}) \\ \dot{y} = p - x - \frac{3y}{4} \end{cases} \quad (1)$$

with  $p$  a positive parameter, we can look for a value of  $p$  such that the system is convergent. To this extent, the expression of the predicate  $R(p, X)$  would be:  $\dot{x} = \dot{y} = 0$ , i.e.  $10(y + x - \frac{x^3}{3}) = p - x - \frac{3y}{4} = 0$ . The DSE implies to look for a value of  $p > 0$  such that  $\exists(x, y)$  such that  $10(y + x - \frac{x^3}{3}) = p - x - \frac{3y}{4} = 0$ .

SMT solvers require well-designed predicates on the qualitative states and frontiers and allow us to check composed properties containing conjunctions and disjunctions, meaning that it is either possible to operate on the state space by giving the satisfiability for a valuation (complete or partial) of  $P$ , or on the design space to evaluate the pertinence of a choice of  $P$ . Our motivation in this work was to apply qualitative reasoning and analysis to the simulation of cyber-physical systems. Therefore, our motivation is to develop our methodology to be able to use qualitative reasoning for DSE to choose or optimize the values of the parameters  $P$  to satisfy some constraints.

## 2.2 Design Space Exploration

Design Space Exploration aims to find a design satisfying a set of constraints  $Const$  or optimizing some utility function  $f$ . Depending on the situation, the formulation will have the form of a satisfiability problem or an optimization problem. Since we want to explore the possibility of using qualitative reasoning, we mainly focused on the satisfiability case. Therefore, the problem takes the form:

$$\begin{cases} \max(V_P = (V_1, V_2, \dots, V_m) \subset \mathbf{D}_P) \\ \text{s.t. } Const(V_P) = \mathbf{True} \end{cases}$$

where  $Const(V_P)$  represents the complete set of constraints that the parameters must satisfy. As qualitative reasoning is based on the loss of numerical values, optimization problems are not adapted for this modeling paradigm.

The sought maximum of the subspace  $V_P$  is a maximum regarding inclusion. Getting the maximum valid set regarding the design constraints is essential to compare the allowed values on other criteria afterward. This comparison is at the base of the design optimization (Fuchs and Neumaier, 2010) and requires further analysis steps. They will not be treated in this article.

The contribution presented here aims to study the possibility of linking the DSE problem with qualitative reasoning, as already mentioned in (Lattmann et al., 2014). Our goal is to open the capabilities of qualitative reasoning to broader perspectives and to present a hint at its possible use for more complex applications than state space partitioning. Moreover, since DSE as presented in (Kang et al., 2011) is mainly limited to structural considerations, the addition of qualitative reasoning to the DSE tools may increase the field of resolution with dynamic and behavioral considerations.

## 3 QUALITATIVE REASONING

As explained in the introduction, 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.

**Definition:** We call **qualitative state** of a system the pair  $(m, qs)$  associated with a list of polynomials  $P$  with  $m \in \mathbf{D}_Q$  the current mode and  $qs \in \{+, 0, -\}^{|P|}$  a vector such that  $\forall i \in \llbracket 1, |P| \rrbracket$ ,  $qs[i] = -$  if  $P[i](X) < 0$ ,  $0$  if  $P[i](X) = 0$ , and  $+$  otherwise.

The qualitative states are then represented as an array of  $\{-, 0, +\}$ , 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 of  $P$ . 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 (Tiwari and Khanna, 2002) to deduce the allowed transition directions.

The Lie derivative (Yano, 2020) consists in a vector field differentiation method enabling the directional derivation of a function. For a polynomial  $p$  whose zeros define borders of the system, the Lie derivative corresponding to  $p$  can be written as  $L_X(p) = \sum_{X_i \in X} \frac{\partial p}{\partial X_i} \frac{\partial X_i}{\partial t}$  where the  $X_i$  are the components of  $X$  the set of continuous variables and  $t$  is the time parameter. Determining the sign of this expression requires the ability to fully evaluate  $p$  and the derivatives of all the variables  $X_i$ .

This computation allows the creation of a qualitative model, where the states and evolution are not represented with numerical values but with variable signs  $(+, 0, -)$  and variation directions (corresponding to the sign of the associated derivative).

As the 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.

We anticipate that qualitative reasoning may improve the DSE efficiency and applicability to complex systems as qualitative reasoning applied on early stages of the design of a system can link the found solutions to non valuated variables. It can apply purely symbolic solving to estimate adapted values of  $P$  to different use cases of a same system depending on the anticipated variations in the state space evolution. Moreover, the abstraction of the state space may allow to propose local solutions to adapt the design of the required qualitative behaviors.

## 4 PARAMETERS EVALUATION

The second important aspect that is necessary for DSE is an analysis method. When a value or a subset of values is available for  $P$ , it is essential to test its pertinence using a solver adapted for the equation structures. Here, as we made the hypothesis of systems defined with polynomial equations and predicates, we use a symbolic polynomial solver. Many behavioral

properties of a dynamic system can be conveniently represented using temporal logic (Han and Sanfelice, 2020). 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  $(m_1, s_1) \implies \forall b \in B(m_1, s_1) \setminus B(m_1, s_2), L_X(p_b) * p_b(s_1) > 0 \wedge \text{for } b_f \in B(m_1, s_1) \cap B(m_1, s_2), L_X(p_{b_f}) * p_{b_f}(s_1) < 0$  where  $B(m_1, s_i)$  is the set of borders defining the qualitative state  $s_i$  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 design space exploration. 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.

## 5 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 de-

pending on the nature of the Design space.

## 5.1 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 (Palesi and Givargis, 2002), machine learning methods (Motamedi et al., 2016), 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  $j^{\text{th}}$  authorised value of the  $i^{\text{th}}$  element of  $P$ . By cutting the branch of the non-satisfiable partial combinations of  $P_{1\dots i}$ , we save the computation time that could be lost to test trivially impossible solutions. This algorithm is detailed in algorithm 1, 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  $p_i$ .

## 5.2 Infinite Design Space

Although the exploration of 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 in 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 2, 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

```

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
end
return possibleConfig

```

Algorithm 1: Branch-and-Bound.

criteria on  $\max(\|a - b\|)_{a,b \in set}$  to stop the splitting process. As in algorithm 1, *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 2 and algorithm 1 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.

## 5.3 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

```

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
     $newConstr = toConstr(p \in set)$ 
     $ConstrTot = newConstr \wedge Constr$ 
     $s = SMT\ solver$ 
     $s.addConstraint(ConstrTot)$ 
     $an = s.Check()$ 
    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 2: Dichotomial search.

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 (Blanchard et al., 2018), 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 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 better corresponds to the selection of the variable's initial values than to parameters valuation.

## 6 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 intend to implement this in the following weeks to put our propositions to the test. Specifically, we plan 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, and on more complex ones such as the Lorenz system, with positive parameters  $\sigma$ ,  $\rho$  and  $\beta$  and state variables  $x$ ,  $y$ , and  $z$ , whose dynamics are given by Equation 1.

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

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases} \quad (3)$$

Both systems exhibit different behaviors depending on the value of their parameters. The convergence or periodicity of such systems being an essential element in the trace of a system, 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 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 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 studied tools allowed this type of resolution but only for low-degree polynomial functions. There is no possible generalization for now.

## 8 RELATED WORKS

Some works have studied the different possibilities offered by qualitative reasoning. (Medimegh et al., 2018) focused on the description and the prediction of the behavior of systems at a high level of abstraction, while (Zaatiti et al., 2018) developed the diagnosis aspect of qualitative reasoning using more numerical methods. (Gueuziec et al., 2023) proposed a technique to automatize system abstraction and the creation of hybrid automata. All these projects focused on the state space study and did not delve into state space partitioning but showed the advantages and the limits of the different approaches. Many works proposed very different approaches regarding design space exploration, from the classic search in a finite design space (Lattmann et al., 2014) to works presenting deep-learning-based strategies (Motamedi et al., 2016). Methods also exist to deal with continuous sets using probability density functions (Blanchard et al., 2018) 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 (Wolff et al., 2014) 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 (Fuchs and Neumaier, 2010; Wang and Shan, 2006).

## 9 CONCLUSION

Theoretically, qualitative reasoning should have an interesting relation with DSE, and we can take advantage of it to optimize performances and computation time. Qualitative abstraction allows a permissive representation of models and a satisfying logical expression of essential system properties. Using an SMT solver on these first-order logic predicates is a relevant method to test the constraints' validity for a specific valuation of the parameter list  $P$ .

## REFERENCES

- Aiguier, M., Bretaudeau, F., and Krob, D. (2010). *Complex Systems Design & Management: Proceedings of the First International Conference on Complex Systems Design & Management CSDM 2010*. Springer Science & Business Media.
- Blanchard, J.-B., Damblin, G., Martinez, J.-M., Arnaud, G., and Gaudier, F. (2018). The uranie platform: an open-source software for optimisation, meta-modelling and uncertainty analysis. *arXiv preprint arXiv:1803.10656*.
- Forbus, K. D. (2014). Qualitative reasoning.
- Fuchs, M. and Neumaier, A. (2010). Discrete search in design optimization. In *Complex Systems Design & Management: Proceedings of the First International Conference on Complex System Design & Management CSDM 2010*, pages 113–122. Springer.
- Gueuziec, B., Gallois, J.-P., and Boulanger, F. (2023). Qualitative reasoning and cyber-physical systems: abstraction, modeling, and optimized simulation. In *MoDeVva 2023-20th workshop on model driven engineering, verification and validation*.
- Han, H. and Sanfelice, R. G. (2020). Linear temporal logic for hybrid dynamical systems: Characterizations and sufficient conditions. *Nonlinear Analysis: Hybrid Systems*, 36:100865.
- Kang, E., Jackson, E., and Schulte, W. (2011). 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*, pages 33–54. Springer.
- Kim, R. G., Doppa, J. R., and Pande, P. P. (2018). Machine learning for design space exploration and optimization of manycore systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6. IEEE.
- Kuipers, B. (1986). Qualitative simulation. *Artificial intelligence*, 29(3):289–338.
- Lattmann, Z., Pop, A., De Kleer, J., Fritzson, P., Janssen, B., Neema, S., Bapty, T., Koutsoukos, X., Klenk, M., Bobrow, D., et al. (2014). Verification and design exploration through meta tool integration with openmodelica. In *Proceedings of the 10th International Modelica Conference*, pages 353–362.
- Medimegh, S., Pierron, J.-Y., and Boulanger, F. (2018). Qualitative simulation of hybrid systems with an application to sysml models. In *MODELWARD*, pages 279–286.
- Motamedi, M., Gysel, P., Akella, V., and Ghiasi, S. (2016). Design space exploration of fpga-based deep convolutional neural networks. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 575–580. IEEE.
- Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. (2018). Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34.
- Palesi, M. and Givargis, T. (2002). Multi-objective design space exploration using genetic algorithms. In

- Proceedings of the tenth international symposium on Hardware/software codesign*, pages 67–72.
- Schafer, B. C. and Wang, Z. (2019). High-level synthesis design space exploration: Past, present, and future. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2628–2639.
- Sugeno, M. and Yasukawa, T. (1993). A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on fuzzy systems*, 1(1):7.
- Tiwari, A. and Khanna, G. (2002). Series of abstractions for hybrid automata. In *International Workshop on Hybrid Systems: Computation and Control*, pages 465–478. Springer.
- Wang, G. G. and Shan, S. (2006). Review of metamodeling techniques in support of engineering design optimization. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 4255, pages 415–426.
- Wolff, E. M., Topcu, U., and Murray, R. M. (2014). Optimization-based trajectory generation with linear temporal logic specifications. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5319–5325. IEEE.
- Yano, K. (2020). *The theory of Lie derivatives and its applications*. Courier Dover Publications.
- Zaatiti, H., Ye, L., Dague, P., Gallois, J.-P., and Travé-Massuyès, L. (2018). Abstractions refinement for hybrid systems diagnosability analysis. In *Diagnostics, Security and Safety of Hybrid Dynamic and Cyber-Physical Systems*, pages 279–318. Springer.

