

Boston University

OpenBU

<http://open.bu.edu>

Boston University Theses & Dissertations

Boston University Theses & Dissertations

2020

Formal methods for partial differential equations

<https://hdl.handle.net/2144/41039>

"Downloaded from OpenBU. Boston University's institutional repository."

BOSTON UNIVERSITY
COLLEGE OF ENGINEERING

Dissertation

**FORMAL METHODS FOR PARTIAL DIFFERENTIAL
EQUATIONS**

by

FRANCISCO PENEDO ALVAREZ

B.S., Universidad Autonoma de Madrid, 2014

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2020

© 2020 by
FRANCISCO PENEDO ALVAREZ
All rights reserved

Approved by

First Reader

Calin Belta, PhD
Professor of Mechanical Engineering
Professor of Systems Engineering
Professor of Electrical and Computer Engineering

Second Reader

Wenchao Li, PhD
Assistant Professor of Electrical and Computer Engineering
Assistant Professor of Systems Engineering

Third Reader

Ioannis Ch. Paschalidis, PhD
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Biomedical Engineering
Professor of Computing and Data Sciences

Fourth Reader

Harold Park, PhD
Professor of Mechanical Engineering
Professor of Materials Science and Engineering

It works! It works! I finally invented something that works! Dr. Emmet
“Doc” Brown, Back to the Future (1985)

Acknowledgments

First and foremost I would like to thank my advisor Calin Belta and my unofficial co-advisor Harold Park for the opportunity they gave me to work with them and all their help and support throughout these years. I would also like to thank the rest of my dissertation committee, Yannis Paschalidis, Wenchao Li and Christos Cassandras for their time and valuable suggestions.

I want to thank all of my colleagues in the BU Robotics Lab, past and present, with special mention to my closest collaborators Christi, Giuseppe and Sadra. I would like to extend my gratitude to everyone else in the Division of Systems Engineering at BU for all their hard work, and special thanks to Elizabeth for her extra help and support.

Finally, I want to thank my family and friends, specially Juan and Mario, who at some point or another have contributed ideas or suggestions, and Maria Ines for her unconditional love and support.

FORMAL METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS

FRANCISCO PENEDO ALVAREZ

Boston University, College of Engineering, 2020

Major Professor: Calin Belta, PhD

Professor of Mechanical Engineering

Professor of Systems Engineering

Professor of Electrical and Computer Engineering

ABSTRACT

Partial differential equations (PDEs) model nearly all of the physical systems and processes of interest to scientists and engineers. The analysis of PDEs has had a tremendous impact on society by enabling our understanding of thermal, electrical, fluidic and mechanical processes. However, the study of PDEs is often approached with methods that do not allow for rigorous guarantees that a system satisfies complex design objectives. In contrast, formal methods have recently been developed to allow the formal statement of specifications, while also developing analysis techniques that can guarantee their satisfaction by design. In this dissertation new design methodologies are introduced that enable the systematic creation of structures whose mechanical properties, shape and functionality can be time-varying.

A formal methods formulation and solution to the tunable fields problem is first introduced, where a prescribed time evolution of the displacement field for different spatial regions of the structure is to be achieved using boundary control inputs. A spatial and temporal logic is defined that allows the specification of interesting properties in a user-friendly fashion and can provide a satisfaction score for the designed

inputs. This score is used to formulate an optimization procedure based on Mixed Integer Programming (MIP) to find the best design.

In the second part, a sampling based assumption mining algorithm is introduced, which is the first step towards a divide and conquer strategy to solve the tunable fields problem using assume-guarantee contracts. The algorithm produces a temporal logic formula that represents initial conditions and external inputs of a system that satisfy a goal given as a temporal logic formula over its state. An online supervised learning algorithm is presented, based on decision tree learning, that is used to produce a temporal logic formula from assumption samples.

The third part focuses on the tunable constitutive properties problem, where the goal is to create structures satisfying a stress-strain response by designing their geometry. The goal is represented as a logic formula that captures the allowed deviation from a reference and provides a satisfaction score. An optimization procedure is used to obtain the best geometric design.

Contents

1	Introduction	1
1.1	Control Synthesis for Partial Differential Equations from Spatio-Temporal Specifications	4
1.2	Data Classification Using Signal Temporal Logic	5
1.3	Assume-Guarantee Contracts and Assumption Mining	7
1.4	Constitutive Property Design for Meta-Materials	8
1.5	Contributions	10
2	Control Synthesis for Partial Differential Equations from Spatio-Temporal Specifications	11
2.1	Preliminaries	11
2.1.1	Finite Element Method	11
2.1.2	Signal Temporal Logic	13
2.2	Signal Temporal Logic for PDEs	16
2.3	Problem Formulation and Approach	17
2.4	Discretization of STL for PDEs	19
2.5	MILP Formulation of Control Synthesis	26
2.6	Extension to Multi-Dimensional PDEs	28
2.7	Extension to Non-Linear Equations	30
2.8	Verification for Sets of Initial Values and Boundary Conditions	32
2.9	Case Study	33
2.9.1	Heat Equation	33

2.9.2	Verification Problem For Heat Equation	34
2.9.3	Elastic Wave Propagation	35
2.9.4	2D Beam	39
2.9.5	Nonlinear Elastic Wave Propagation	41
3	A Decision Tree Approach to Data Classification using Signal Temporal Logic	44
3.1	Parametric Signal Temporal Logic	44
3.2	Problem formulation	45
3.3	Learning decision trees	45
3.3.1	Parameterized learning algorithm	46
3.3.2	Tree to STL formula	48
3.3.3	PSTL primitives	49
3.3.4	Impurity measures	51
3.3.5	Stop conditions	55
3.3.6	Complexity	55
3.4	Case Studies	56
3.4.1	Maritime surveillance	56
3.4.2	Fuel control system	58
3.5	Implementation and results	59
3.5.1	Maritime surveillance	61
3.5.2	Fuel control	63
4	Assumption Mining	65
4.1	Assume-Guarantee Contracts and Assumption Mining	65
4.2	Online Decision Tree STL Inference	67
4.3	Sampling Based Assumption Mining	68
4.4	Case Study	70

5	Constitutive Property Design	73
5.1	Stress-Strain Curves and Logics	73
5.2	Problem Formulation and Solution	78
5.3	Case Study	79
6	Conclusions and Future Work	82
6.1	Future Research Directions	84
	References	86
	Curriculum Vitae	93

List of Tables

3.1 STL inference algorithm meta-parameters	61
---	----

List of Figures

2·1	Summary of the PDE discretization approach	19
2·2	Results for boundary control of 1D first order PDE	35
2·3	Results for verification of 1D first order PDE	36
2·4	Results for boundary control of 1D second order PDE with a complex specification	38
2·5	Results for boundary control of 1D second order PDE with existential specification	39
2·6	Results for boundary control of 1D second order PDE with specification over strain	39
2·7	Results for boundary control of 2D second order PDE	41
2·8	Results for boundary control of 1D second order nonlinear PDE	43
3·1	Obtaining an STL formula from a decision tree	49
3·2	Naval surveillance dataset	57
3·3	Fuel control dataset	60
3·4	Sample of the naval surveillance dataset	62
4·1	Assumption mining results	72
5·1	Typical s-s curve	75
5·2	Shape of the smooth score against the basic score of a predicate with different priorities.	77
5·3	Proposed workflow for Stress-Strain curve verification.	77

5.4	Results for constitutive design.	81
-----	--	----

List of Abbreviations

AGC	Assume-Guarantee Contract
FEM	Finite Element Method
IBVP	Initial Boundary Value Problem
MILP	Mixed Integer Linear Program
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PSTL	Parametric Signal Temporal Logic
S-S	Stress-Strain
S-STL	Spatial Signal Temporal Logic
SSPL	Stress-Strain Predicate Logic
STL	Signal Temporal Logic
TLI	Temporal Logic Inference

Chapter 1

Introduction

Partial differential equations (PDEs) model nearly all of the physical systems and processes of interest to scientists and engineers. Some well-known examples include the Navier-Stokes equation for fluid mechanics, the Maxwell equations for electromagnetics, the Schrödinger equation for quantum mechanics, the heat equation and the wave equation. The analysis of these and other PDEs has had a tremendous impact on society by enabling our understanding of thermal, electrical, fluidic and mechanical processes.

While a mature field, the study of PDEs is often approached through simulations in which approximate models obtained through spatio-temporal discretization techniques, such as the Finite Element Method (FEM) (Hughes, 2000), are solved numerically. These methods, however, do not allow for rigorous guarantees that a system satisfies a complex specification. Other approaches generalizing classic Ordinary Differential Equations (ODEs) tools such as Lyapunov analysis and backstepping (Krstic and Smyshlyaev, 2008) do provide some guarantees and can be used to obtain boundary control strategies for a wide variety of PDEs. However, they are restricted to classical control objectives such as stabilization and cost optimization (Meurer and Kugi, 2009).

The formal statement of specifications and the development of analysis techniques that can guarantee their satisfaction by design has been the main focus of the formal methods field. During the past decades, many specification languages have been de-

fined, such as Linear Temporal Logic (LTL) (Gerth et al., 1996) and Signal Temporal Logic (STL) (Donzé and Maler, 2010). Originally, these logics have been applied to the study of finite systems, although more recently abstraction procedures have been developed to reduce problems with ODEs to finite models (for example through state space discretization (Kloetzer and Belta, 2008) or mixed-integer linear program (MILP) encodings (Raman et al., 2014)). However, these techniques cannot be immediately applied to the analysis of PDEs due to the lack of spatial information in the formal language. This issue was first addressed in (Grosu et al., 2009) in the context of spatially distributed systems, which can be viewed as PDEs in a discretized spatial domain. In their work, the authors view the system state as an image and define a formal language with explicit spatial information called Linear Superposition Logic (LSSL) based on quadtrees, a tree-based abstraction of the system. In (Bartocci et al., 2016), a variant of LSSL with quantitative semantics, Tree Spatial Superposition Logic (TSSL), is used for (steady-state) pattern synthesis in reaction diffusion systems, while in (Haghighi et al., 2015), a new formal language combining TSSL and STL, Spatial Temporal Logic (SpaTeL), allows the synthesis of dynamical patterns. The specification of patterns in these logics is difficult, however, and machine learning techniques are needed in order to obtain logic formulas corresponding to a set of examples of the desired pattern, which is not always available or desirable for some applications where the system behavior must be specified a priori. More recently, a new spatio-temporal logic called STREL was introduced in (Ezio Bartocci et al., 2017) to specify the behavior of mobile and spatially distributed Cyber-Physical Systems.

One of the major drawbacks of these methods is the issue of scalability. Both of the main approaches, automata and optimization, quickly become unfeasible when the system dimension increases, which poses a problem when dealing with infinite-

dimensional systems such as PDEs. In the case of optimization based techniques, a recent trend has focused on decentralized synthesis and verification where the coupling between subsystems is formally captured in Assume-Guarantee Contracts (AGCs) (Sadraddini et al., 2017). However, the key assumption of monotonicity is not present in PDE systems.

Besides classical problems that focus on the temporal evolution of the state of the system, PDEs allow for a different kind of problems where the aim is to specify the intrinsic behavior of the system through its so called constitutive response. In this case, instead of designing boundary control inputs, the engineer is tasked with the design of the geometry of a structure that exhibits the desired mechanical or thermal properties (Bertoldi et al., 2008; Shim et al., 2013a). The traditional approach to this problem, topology optimization (Bendsoe, 1989; Bendsoe and Sigmund, 1999; Clausen et al., 2015; Deaton and Grandhi, 2014; Frei et al., 2005; Maute et al., 2015; Osanov and Guest, 2016; Sigmund and Torquato, 1999; Sigmund and Maute, 2013; van Dijk et al., 2013; Nanthakumar et al., 2015; Nanthakumar et al., 2016; Nanthakumar et al., 2017; Ghasemi et al., 2017; Allaire et al., 2004; Wang et al., 2003; Picelli et al., 2018), is not enough to tackle both geometric and material nonlinearity.

In this dissertation, we present the first formal methods approach to the design of PDE systems. We first consider boundary control problems with spatio-temporal specifications over the state of the system. Then, we make a start on tackling the scalability issues of our approach. In the last part of the dissertation, we focus on the design of structures and materials from specifications over their intrinsic mechanical response.

1.1 Control Synthesis for Partial Differential Equations from Spatio-Temporal Specifications

In the first part of the dissertation, we extend STL to allow specifications over the solutions of PDEs in such a way that both temporal and spatial properties can be formally stated in a user-friendly manner. We call the resulting language Spatial-STL or S-STL. Then, we formulate and solve the problem of synthesizing a boundary control input for a PDE such that a property defined in S-STL is satisfied.

Our approach uses the FEM to approximate the trajectory of the PDE by converting the PDE, as is standard with the FEM (Hughes, 2000), to a system of ODEs. The FEM is a well-established numerical method to obtain approximate numerical solutions to PDEs, and is particularly well-suited to problems with complicated geometries where exact solutions cannot be found. The state space of the resulting ODE system represents the field values of the PDE at discrete locations, called nodes, obtained after discretizing the domain. In the process of approximating the PDE trajectory by the FEM model and then discretizing it both in space (considering only the values at the nodes) and time, we define a conservative reformulation of the specification that follows the same approximation and discretization steps. In order to account for the approximation and discretization errors, we introduce correction terms in the formula in such a way that all trajectories satisfying the corrected formula also satisfy the original one. This procedure is similar to the syntactical re-writing rules for MTL proposed in (Abbas et al., 2014) in order to automatically infer properties satisfied in a derived system model (via simplification or implementation, for example) from properties satisfied in the original model. Finally, we encode the resulting control problem into a MILP following previously established methods.

This work is an extension of the conference version (Penedo et al., 2018), where we presented the control synthesis problem for 1D linear PDEs. Here, we extend

our framework to multidimensional PDEs and to allow some non-linearities in the PDE, in particular non-linear constitutive relations that can be approximated well with piecewise linear functions with few interpolation points. We also present a corresponding verification problem and solve it using the same tools developed for the synthesis problem. The performance of our approach has been greatly improved by including an optional presolving step where an approximation to the MILP solution is obtained first using a gradient-free optimization method and used as a starting point for the MILP. This allows us to avoid in practice the exponential complexity of the MILP solver.

1.2 Data Classification Using Signal Temporal Logic

In the second part of the dissertation, we momentarily deviate from the PDE theme and introduce a framework for the inference of timed temporal logic properties from data. The relevance to the rest of the dissertation comes from its use as part of a divide and conquer approach to the control synthesis problem for PDEs already mentioned.

One of the main problems in machine learning is the so called *two-class classification problem*. In this setting, the goal is to build a classifier that can distinguish objects belonging to one of two possible classes. This problem is of fundamental importance because its solution leads to solving the more general multi-class problem (Ripley, 1996). Furthermore, it can be directly used in the context of anomaly detection, where the objective is to find patterns in data that do not conform to the expected behavior. These non-conforming patterns are often referred to as *anomalies* or *negatives*, whereas the normal working conditions are usually referred to as *targets* or *positives*. Given the importance of this problem and its broad applicability, it has been the topic of several surveys (Isermann, 2006; Chandola et al., 2009).

A specific formulation of the two-class problem is determined by several factors such as the nature of the input data, the availability of labels, as well as the constraints and requirements determined by the application domain (Chandola et al., 2009). In this work, we deal with data in form of finite time series, called signals or traces, and we suppose that the labels of these traces are available. That is, the true class of each trace is known, either *positive* or *negative*, and this information is exploited during the classifier construction phase (*supervised learning*). We tackle the two-class classification problem by bringing together concepts and tools from formal methods and machine learning. Our thesis is that a *formal specification* of the normal working conditions can be gleaned directly from execution traces and expressed in the form of STL formulas. The inferred formulae can then be applied directly as data classifiers for new traces. In this context, some work has been initially done to optimize the parameters of a formula for a given, fixed, formula structure (Jin et al., 2015; Asarin et al., 2012; Yang et al., 2012). Kong et. al. (Kong et al., 2014; Jones et al., 2014) were the first to propose an algorithm to learn *both* the formula structure and its parameters from data and called this approach *temporal logic inference* (TLI). This approach, while retaining many qualities of traditional classifiers, presents several additional advantages. First, STL formulas have precise meaning and allow for a rich specification of the normal behaviour that is easily *interpretable by humans*. Second, anomaly detection methods commonly applied to time series data are often model-based, i.e., they require a *good* model of the system running alongside the physical system (Isermann, 2006). Third, classical machine learning methods are often over specific to the task. That is, they focus exclusively on solving the classification problem but offer no other insight on the system where they have been applied. On the contrary, TLI fits naturally as a step in the system’s design workflow and its analysis and results can be employed in other phases.

In this work, which we presented in (Bombara et al., 2016), we propose a decision-tree based framework for solving the two-class classification problem involving signals using STL formulas as data classifiers. We refer to it as *framework* because we are not just proposing a single algorithm but a *class* of algorithms. Every algorithm produces a binary decision tree which can be translated to an STL formula and used for classification purposes. Each node of a tree is associated with a simple formula, chosen from a finite set of primitives. Nodes are created by finding the best primitive, along with its optimal parameters, within a greedy growing procedure. The optimality at each step is assessed using *impurity* measures, which capture how well a primitive splits the signals in the training data. The impurity measures described here are modified versions of the usual impurity measures to handle signals, and were obtained by exploiting the *robustness degree* concept (Donzé and Maler, 2010). Our novel framework presents several advantages. In particular, the proposed incremental construction procedure requires the optimization of a small and fixed number of primitives at each node. Moreover, the number of objects to be processed decreases at each iteration. These two features greatly improve the execution time and the accuracy compared to the algorithms proposed in (Kong et al., 2014; Jones et al., 2014).

1.3 Assume-Guarantee Contracts and Assumption Mining

In order to address the scalability issues of our optimization based approach discussed in the first part, we focus on adapting existing AGC based approaches to distributed control of monotone systems (Sadraddini et al., 2017) to relax the monotonicity assumption. As a first step towards an AGC based approach to distributed control of non-monotone systems, we propose a sampled based assumption mining algorithm that leverages the STL inference framework introduced earlier. Assumption mining is

by itself an interesting problem that can provide the means to identify failure modes and determine the limits of safe operation of a system. In this case, the acceptable behavior of the system is encoded as a logical formula and we seek a formula describing the largest set of initial states and external disturbances for which the system satisfies the specification. The problem has been previously studied for the synthesis of discrete controllers from temporal specifications (Li et al., 2011) and in the particular case of monotone systems (Kim et al., 2016).

We build an approximation of the assumption set iteratively using a process similar to (Kim et al., 2016), with the current approximation represented as an STL formula inferred from samples. The precision of the approximation can be controlled by a parameter. We prove the algorithm finishes in finite time and the resulting formula can be used to reject all invalid assumptions at the cost of being conservative.

1.4 Constitutive Property Design for Meta-Materials

Imagine the ability to, given a user-specified mechanical behavior, systematically pattern a structure to produce the desired behavior. For example, can we tailor the features of a material to endow it a desired constitutive response? Can we give an elastic material a tunable, effective *yield point*, or make it strain harden at a specified amount of displacement? This response to loading, whether being ductile or brittle, strain hardening or softening, strong or weak, is found in the stress–strain curve, which serves as a fundamental piece of information that describes the global mechanical properties of structures and materials.

All of the above examples fall into a larger category of desired structural mechanical response called *designer matter* (Reis et al., 2015; Bertoldi et al., 2017; Kochmann and Bertoldi, 2017). In designer matter, structural patterning is used to develop structures or metamaterials that exhibit unique properties (negative Poisson’s ratio,

negative compressibility, negative thermal expansion, etc) (Yu et al., 2018). Despite these recent advances, we still lack the systematic ability to design a material or structure that will satisfy arbitrary, user-defined sets of material properties.

Designing structures that provide desired mechanical or material properties in response to applied loads has traditionally been done using topology optimization (Bendsoe, 1989; Bendsoe and Sigmund, 1999; Clausen et al., 2015; Deaton and Grandhi, 2014; Frei et al., 2005; Maute et al., 2015; Osanov and Guest, 2016; Sigmund and Torquato, 1999; Sigmund and Maute, 2013; van Dijk et al., 2013; Nanthakumar et al., 2015; Nanthakumar et al., 2016; Nanthakumar et al., 2017; Ghasemi et al., 2017; Allaire et al., 2004; Wang et al., 2003; Picelli et al., 2018), which attempts to determine the optimal geometry and material distribution of a body subject to well-defined constraints, or design variables. While topology optimization has been widely used to design structures with pre-determined properties, it is best-suited for determining a fixed structure that exhibits static, time-independent properties based on linear elastic material properties. Other works have developed topology optimization techniques to account for geometric (Gea and Luo, 2001; Bruns and Tortorelli, 2001; Bruns and Tortorelli, 2003; Wang et al., 2014; Buhl et al., 2000) and material (Maute et al., 1998; Swan and Kosaka, 1997; Jung and Gea, 2004; Zhang et al., 2017) nonlinearity, and also inertial effects (Alberdi and Khandelwal, 2019; Filipov et al., 2016; Lavan, 2019; Nakshatrala and Tortorelli, 2015). However, it remains a significant challenge to use topology optimization to design a structure, that exhibits a prescribed response that must account for both geometric and material nonlinearity.

We propose a formal methods framework to formulate and solve the problem of specifying the constitutive response of a structure and synthesizing its geometry so that the specification is satisfied by design. We introduce a predicate logic tailored to the description of interesting constitutive properties of a material, as well as show how

to adapt S-STL for the same purpose. Both logics come equipped with a quantitative robustness score that allows us to formulate an optimization procedure where different geometric designs are modeled, simulated and scored, and a gradient-free optimization algorithm is used to select the best design.

1.5 Contributions

In conclusion, the contributions of the dissertation are the following. In the first part of the dissertation, Chapter 2, we propose a formal framework for the verification and synthesis of boundary control inputs for PDE systems from spatio-temporal specifications. We applied this framework to 1D and 2D linear PDEs, as well as 1D non-linear PDEs. In the second part of the dissertation, we focus on the first steps towards a divide and conquer approach to the control synthesis problem of the first part. In Chapter 3, we propose a decision tree approach to STL inference from labeled sampled data and show its performance on the maritime security and automotive fields. The inference algorithm is used in Chapter 4, where we present a sampled based assumption mining algorithm for discrete time linear systems. We do not assume monotonicity of the system, which makes the algorithm applicable to discretizations of PDE systems. Finally, in Chapter 5 we present a formal methods framework for the geometric design of structures from given constitutive response specifications.

Chapter 2

Control Synthesis for Partial Differential Equations from Spatio-Temporal Specifications

In this chapter, we develop a framework for the verification and synthesis of boundary control inputs in PDE systems from specifications that describe the temporal and spatial behavior of the system. First, we define an appropriate formal language to describe such specifications. Then, we reformulate the problem into an optimization problem over discrete time finite dimensional systems. Finally, we encode the problem as a Mixed Integer Linear Program (MILP).

Our framework is conservative, i.e., a valid solution may not be found if it is too close to violating the specification. In order to find an MILP encoding, we require linearity of the PDE, although we show how this can be relaxed for some forms of nonlinearity. Examples in 1D, 2D, first and second order, and nonlinear PDEs illustrating the performance of the framework are included.

2.1 Preliminaries

2.1.1 Finite Element Method

In this section we provide a summary of the Finite Element Method (FEM) (Hughes, 2000). For simplicity, we present the method applied to a heat equation over a one-dimensional domain, although the same technique can be applied to other PDEs.

Let $\Omega = (0, L) \subset \mathbb{R}$ be an open interval representing the interior of a one-dimensional rod of length L ; $\rho, c, \kappa > 0$ constants denoting density, capacity and conductivity of the rod's material respectively; $g = (g_0, g_L) \in \mathbb{R}^2$ the (time-independent) boundary conditions at each end of the rod; and $u_0 : \Omega \rightarrow \mathbb{R}$ an initial value for the temperature on the rod. The evolution of the temperature at each point in the rod can be described by a function $u : \bar{\Omega} \times [0, T] \rightarrow \mathbb{R}$, where $T > 0$ denotes the final time and can be infinity and $\bar{\Omega}$ is the closure of Ω (which we call the spatial domain), such that the following initial boundary value problem (IBVP) is satisfied:

$$\Sigma^H(u_0, g) \left\{ \begin{array}{l} \rho c \frac{\partial u}{\partial t} - \kappa \frac{\partial^2 u}{\partial x^2} = 0, \text{ on } \Omega \times (0, T), \\ u(0, t) = g_0, \forall t \in (0, T), \\ u(L, t) = g_L, \forall t \in (0, T), \\ u(x, 0) = u_0(x), \forall x \in \Omega. \end{array} \right. \quad (2.1)$$

An important aspect of the FEM is the creation of a weak formulation of the governing PDEs in (2.1). Its purpose is to reduce the second order spatial derivative of u in (2.1) to first derivatives so that low order (in this case, linear) polynomials can be used to approximate the field value u . Let D be the set of sufficiently smooth real-valued functions on $\bar{\Omega} \times (0, T)$ such that all $u \in D$ satisfy $u(0, t) = g_0, u(L, t) = g_L, \forall t \in (0, T)$, and V a similar set of time independent functions such that all $w \in V$ satisfy $w(0) = w(L) = 0$. The problem is to find $u \in D$ such that for all $w \in V$,

$$\begin{aligned} \int_{\Omega} \frac{\partial w}{\partial x} \kappa \frac{\partial u}{\partial x} d\Omega + \int_{\Omega} w \rho c \frac{\partial u}{\partial t} d\Omega &= 0, \\ \int_{\Omega} w \rho c u(\cdot, 0) d\Omega &= \int_{\Omega} w \rho c u_0 d\Omega. \end{aligned} \quad (2.2)$$

We now obtain an approximate solution to the weak formulation by considering (2.2) with u and w in subspaces of D and V . Let $\{x_i\}_{i=0}^{n+1}$, where $x_0 = 0, x_{n+1} = L, x_i \in$

$\Omega, i = 1, \dots, n$, be a partition of $\bar{\Omega}$. Let $d_i(t), i = 0, \dots, n + 1$ represent the temperature of the rod at node x_i , with $d_0(t) = g_0, d_{n+1} = g_L$ and let $d = (d_1, \dots, d_n)' \in \mathbb{R}^n$. We define the following linear node shape function matrices for $i = 0, \dots, n + 1$:

$$N_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & i > 0, x \in [x_{i-1}, x_i], \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & i < n + 1, x \in [x_i, x_{i+1}], \end{cases} \quad (2.3)$$

which results in the following linear interpolation of the field variable u in terms of its nodal values d :

$$u^d(x, t) = \sum_{i=0}^{n+1} N_i(x) d_i(t). \quad (2.4)$$

Consider the subspaces $D^h \subset D$ and $V^h \subset V$ of linear interpolations defined above and time-invariant interpolations respectively. It can be shown that $u^d(x, t)$ is a solution of the weak formulation over the sets D^h and V^h , where d evolves according to the following linear system:

$$\Sigma_{FEM}^H(u_0, g) \begin{cases} M\dot{d} + Kd = F(g), \\ d_i(0) = u_0(x_i), \quad i = 1, \dots, n. \end{cases} \quad (2.5)$$

In the above, M, K and F are the mass, stiffness and external force matrices respectively, whose specific form depend on the partition and the parameters of the PDE. In general, M can be considered diagonal and K is a banded matrix, in this specific PDE having bandwidth 3. In general, we will denote a PDE by $\Sigma(\dots)$ and we will call $\Sigma_{FEM}(\dots)$ the FEM system corresponding to a PDE system $\Sigma(\dots)$.

2.1.2 Signal Temporal Logic

Let \mathbb{R} be the set of real numbers. For $t \in \mathbb{R}$, we denote the interval $[t, \infty)$ by $\mathbb{R}_{\geq t}$. We use $\mathcal{S} = \{s : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n\}$ with $n \in \mathbb{N}$ to denote the set of all continuous parameterized curves in the n -dimensional Euclidean space \mathbb{R}^n . In this paper, an

element of \mathcal{S} is called a *signal* and its parameter is interpreted as *time*. Given a signal $s \in \mathcal{S}$, the components of s are denoted by s_i , $i \in \{1, \dots, n\}$. The set \mathcal{F} contains the projection operators from a signal s to one of its components s_i , specifically $\mathcal{F} = \{f_i : \mathbb{R}^n \rightarrow \mathbb{R}, f_i(s) = s_i, i \in \{1, \dots, n\}\}$. The *suffix* at time $t \geq 0$ of a signal is denoted by $s[t] \in \mathcal{S}$ and it represents the signal s shifted forward in time by t time units, i.e., $s[t](\tau) = s(\tau + t)$ for all $\tau \in \mathbb{R}_{\geq 0}$.

The syntax of *Signal Temporal Logic* (STL) (Maler and Nickovic, 2004) is defined as follows:

$$\phi ::= \top \mid p_{f(x) \leq \mu} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathbf{U}_{[a,b]} \phi_2$$

where \top is the Boolean *true* constant; $p_{f(x) \leq \mu}$ is a predicate over \mathbb{R}^n defined by the function $f \in \mathcal{F}$ and $\mu \in \mathbb{R}$ of the form $p_{f(x) \leq \mu}(x) = f(x) \leq \mu$; \neg and \wedge are the Boolean operators negation and conjunction; and $\mathbf{U}_{[a,b]}$ is the bounded temporal operator *until*. We use \perp to denote the Boolean *false* constant.

The semantics of STL is defined over signals in \mathcal{S} recursively as follows (Maler and Nickovic, 2004):

$$\begin{aligned} s[t] \models \top & \Leftrightarrow \top \\ s[t] \models p_{f(x) \leq \mu} & \Leftrightarrow (f(s(t)) \leq \mu) \\ s[t] \models \neg\phi & \Leftrightarrow \neg(s[t] \models \phi) \\ s[t] \models (\phi_1 \wedge \phi_2) & \Leftrightarrow (s[t] \models \phi_1) \wedge (s[t] \models \phi_2) \\ s[t] \models (\phi_1 \mathbf{U}_{[a,b]} \phi_2) & \Leftrightarrow \exists t_u \in [t + a, t + b] \text{ s.t. } (s[t_u] \models \phi_2) \\ & \quad \wedge (\forall t_1 \in [t, t_u] s[t_1] \models \phi_1) \end{aligned}$$

A signal $s \in \mathcal{S}$ is said to satisfy an STL formula ϕ if and only if $s[0] \models \phi$. We extend the type of allowed inequality predicates in STL to $s[t] \models p_{f(x) > \mu} \equiv s[t] \models \neg p_{f(x) \leq \mu}$. Thus, predicates are defined in this paper by a function $f \in \mathcal{F}$, a real number $\mu \in \mathbb{R}$

and an order relation $\sim \in \{\leq, >\}$. The other Boolean operations (i.e., disjunction, implication, equivalence) are defined in the usual way. Also, the temporal operators *eventually* and *globally* are defined as $\mathbf{F}_{[a,b]}\phi \equiv \top \mathbf{U}_{[a,b]}\phi$ and $\mathbf{G}_{[a,b]}\phi \equiv \neg \mathbf{F}_{[a,b]}\neg\phi$, respectively.

In addition to Boolean semantics defined above, STL admits *quantitative semantics* (Donzé and Maler, 2010; Fainekos and Pappas, 2009), which is formalized by the notion of *robustness degree*. The robustness degree of a signal $s \in \mathcal{S}$ with respect to an STL formula ϕ at time t is a function $r(s, \phi, t)$ and is recursively defined as

$$\begin{aligned}
r(s, \top, t) &= r_{\top} \\
r(s, p_{f(x) \leq \mu}, t) &= \mu - f(s(t)) \\
r(s, \neg\phi, t) &= -r(s, \phi, t) \\
r(s, \phi_1 \wedge \phi_2, t) &= \min\{r(s, \phi_1, t), r(s, \phi_2, t)\} \\
r(s, \phi_1 \mathbf{U}_{[a,b]}\phi_2, t) &= \\
&\sup_{t_u \in [t+a, t+b]} \left\{ \min \left\{ r(s, \phi_2, t_u), \inf_{t_1 \in [t, t_u]} \{r(s, \phi_1, t_1)\} \right\} \right\}
\end{aligned}$$

where $b > a > 0$ and $r_{\top} \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ is a large constant representing the maximum value of the robustness. Note that a positive robustness degree $r(s, \phi, 0)$ of a signal s with respect to a formula ϕ implies that s satisfies ϕ (in Boolean semantics). In the following, we denote by $r(s, \phi)$ the robustness degree $r(s, \phi, 0)$ at time 0. Robustness can be extended to the derived predicates and operators as follows:

$$\begin{aligned}
r(s, p_{f(x) > \mu}, t) &= f(s(t)) - \mu \\
r(s, \phi_1 \vee \phi_2, t) &= \max\{r(s, \phi_1, t), r(s, \phi_2, t)\} \\
r(s, \mathbf{F}_{[a,b]}\phi, t) &= \sup_{t_f \in [t+a, t+b]} \{r(s, \phi, t_f)\} \\
r(s, \mathbf{G}_{[a,b]}\phi, t) &= \inf_{t_g \in [t+a, t+b]} \{r(s, \phi, t_g)\}
\end{aligned}$$

Moreover, the interpretation of robustness degree as a quantitative measure of satisfaction is justified by the following proposition from (Donzé et al., 2013).

Proposition 2.1. *Let $s \in \mathcal{S}$ be a signal and ϕ an STL formula such that $r(s, \phi) > 0$. All signals $s' \in \mathcal{S}$ such that $\|s - s'\|_\infty < r(s, \phi)$ satisfy the formula ϕ , i.e., $s' \models \phi$.*

As an example, consider the temperature at the end of a rod to be $s(t) = 2t$, and the specification $\phi = \mathbf{G}_{[1,2]}s > 0.5$. We have $s \models \phi$, since $s(t) > 0.5, \forall t \in [1, 2]$ and robustness $r(\phi, s, 0) = 0.5$ since $\min_{t \in [1, 2]}(s(t) - 0.5) = 0.5$.

2.2 Signal Temporal Logic for PDEs

In order to define specifications over the trajectories of PDEs, we extend regular STL using the following set of predicates: let Λ be a set of predicates over a set $\bar{\Omega}$, where each predicate $\lambda \in \Lambda$ is defined as a tuple $(Q_\lambda, X_\lambda, \mu_\lambda, D_\lambda)$, and represented using the syntax $Q_\lambda x \in X_\lambda : D_\lambda u(x) - \mu_\lambda(x) > 0$, where:

- $Q_\lambda \in \{\forall, \exists\}$ is the spatial quantifier,
- $X_\lambda \subseteq \bar{\Omega}$ is the spatial domain of the predicate and we require it to be a closed set,
- $\mu_\lambda : X_\lambda \rightarrow \mathbb{R}$ is a continuous and differentiable function representing the reference profile, and
- $D_\lambda \in \{\frac{d^i}{dx^i}\}_{i=0,1,\dots}$ is a differential operator, with $\frac{d^0}{dx^0} = id$ the identity.

We consider STL formulas with the usual syntax and semantics over the set of predicates Λ , which we call Spatial-STL (S-STL). The satisfaction of a predicate with respect to a continuous-time signal $u : \bar{\Omega} \times [0, T] \rightarrow \mathbb{R}$ at time $t \in \mathbb{R}$ is defined as $u[t] \models \lambda \iff D_\lambda u(x, t) - \mu_\lambda(x) > 0$ for all $x \in X_\lambda$ if $Q_\lambda = \forall$ or for some $x \in X_\lambda$ otherwise. We define the quantitative semantics as before with $r(\lambda, u, t) =$

$\min_{x \in X_\lambda} (D_\lambda u(x, t) - \mu_\lambda(x))$ if $Q_\lambda = \forall$ and $r(\lambda, u, t) = \max_{x \in X_\lambda} (D_\lambda u(x, t) - \mu_\lambda(x))$ otherwise. For convenience, we define syntactic sugar for predicates with the opposite inequality using the following equivalence: $\forall x \in X : Du(x) - \mu \leq 0 \equiv \neg \exists x \in X : Du(x) - \mu > 0$, and similarly for an existential predicate.

Example 2.1. *Consider a metallic rod of 100 mm. The temperature at one end of the rod is fixed at 300 K, while a heat source is applied to the other end. The temperature of the rod follows a heat equation similar to (2.1). We want the temperature distribution of the rod to be within 3 K of the linear profile $\mu(x) = \frac{x}{4} + 300$ at all times between 4 and 5 seconds in the section between 30 and 60 mm. Furthermore, the temperature should never exceed 345 K at the point where the heat source is applied ($x = 100$). We can formulate such a specification using the following S-STL formula:*

$$\begin{aligned} \phi_{ex} = & \mathbf{G}_{[4,5]} \left((\forall x \in [30, 60] : u(x) - \left(\frac{x}{4} + 303\right) < 0) \wedge \right. \\ & \left. (\forall x \in [30, 60] : u(x) - \left(\frac{x}{4} + 297\right) > 0) \right) \wedge \\ & \mathbf{G}_{[0,5]} (\forall x \in [100, 100] : u(x) - 345 < 0). \end{aligned} \quad (2.6)$$

Throughout this paper, we will use the notation $S(\dots) \models \psi$ to denote that the trajectory of system S (either a PDE system or its corresponding FEM system) with initial value s_0 and boundary conditions g satisfies formula ψ (either in regular STL or in S-STL).

2.3 Problem Formulation and Approach

Let $\partial\Omega$ be the boundary of Ω . We consider $\partial\Omega$ to be partitioned in four regions $\{\partial\Omega_d, \partial\Omega_n, \partial\Omega_D, \partial\Omega_N\}$. Let $u(x, t) : \bar{\Omega} \times [0, T) \rightarrow \mathbb{R}^n$ be the state of a system evolving according to the IBVP $\Sigma(u_0, g_d, g_n, v_D, v_N)$:

$$\Sigma(\dots) \left\{ \begin{array}{l} f(x, t, u, \frac{\partial u}{\partial t}, \frac{\partial u}{\partial x_i}, \dots) = 0, \text{ on } \Omega \times (0, T), \\ u(x, t) = g_d(x, t), \forall x \in \partial\Omega_d, \forall t \in (0, T), \\ \frac{\partial u}{\partial n}(x, t) = g_n(x, t), \forall x \in \partial\Omega_n, \forall t \in (0, T), \\ u(x, t) = v_D(x, t), \forall x \in \partial\Omega_D, \forall t \in (0, T), \\ \frac{\partial u}{\partial n}(x, t) = v_N(x, t), \forall x \in \partial\Omega_N, \forall t \in (0, T). \end{array} \right. \quad (2.7)$$

In the above, f is a function of space, time, state and all state derivatives, n is the normal vector to $\partial\Omega$, g_d and g_n are the Dirichlet and Neumann prescribed boundary conditions respectively, and v_D and v_N are the Dirichlet and Neumann boundary control inputs respectively. We will use Σ to refer to the system given by (2.7), including the partition of $\partial\Omega$, with unspecified initial value and boundary conditions.

Problem 2.1 (Boundary Control Synthesis Problem). *Given a PDE Σ , an initial value u_0 , prescribed boundary conditions g_d and g_n , an S-STL formula ψ over a set of predicates Λ , and admissible control sets V_D and V_N , synthesize control inputs $v_D \in V_D$ and $v_N \in V_N$ such that the trajectory of $\Sigma(u_0, g_d, g_n, v_D, v_N)$ satisfies ψ .*

In the above formulation, the admissible control inputs are given in their most general form as sets of allowed control functions (note that from (2.7), $v_D : \partial\Omega_D \times (0, T) \rightarrow \mathbb{R}^n$ and $v_N : \partial\Omega_N \times (0, T) \rightarrow \mathbb{R}^n$). In practice, they are described in terms of control inputs constraints, such as $v_D(x, t) < 100, \forall x \in \partial\Omega_D, \forall t \in (0, T)$, as well as assumptions on the form of the function, such as piecewise affine in t and x . We also make the following assumptions: the function f in (2.7) is linear (we will relax this assumption in Sec. 2.7), and only a finite number of derivatives of u appear; and the sets V_D and V_N are described as polytopes in the parameters of a linear parameterization of the control inputs.

We solve Problem 2.1 by reformulating it into a synthesis problem for a discrete-time linear system instead. This is done in three steps, described in Sec. 2.4, by

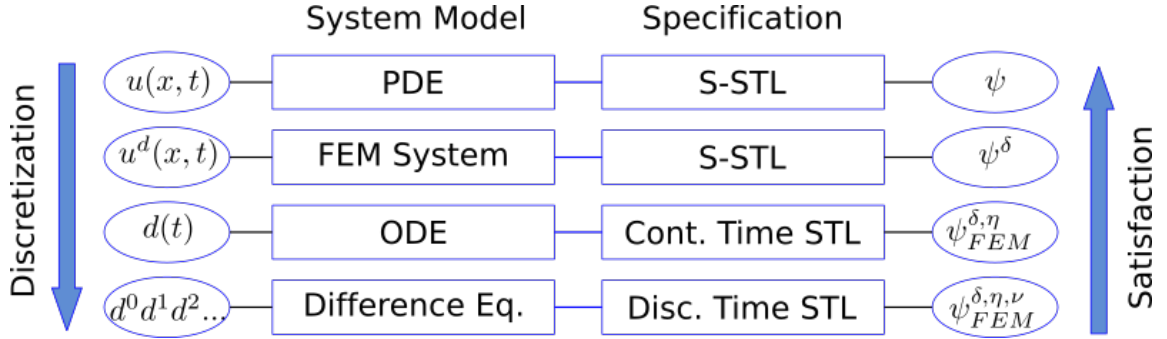


Figure 2-1: Summary of our approach. The PDE model is discretized in three steps to obtain a set of difference equations. The S-STL specification is rewritten following the same steps so that satisfaction in the discretized model is preserved.

considering the FEM approximation to the real solution, a spatial discretization and a temporal discretization. Then, we encode the resulting linear system and specification in a MILP and solve for the control inputs that maximize the robustness with respect to the discretized specification, which is described in Sec. 2.5.

2.4 Discretization of STL for PDEs

In this section we define a series of reformulations of an S-STL formula over Λ such that satisfaction of each successive reformulation guarantees that of the previous one. A diagram showing the hierarchy of the reformulations is shown in Fig. 2-1. The theory developed in this section is stated for a general 1D PDE. In Sec. 2.6 we provide the necessary changes to apply this framework to multi-dimensional PDEs.

We first reformulate the specification ψ so that we can work with the approximation given by Σ_{FEM} . Recall that we denote as $u(x, t)$ the trajectory of Σ^1 and $u^d(x, t)$ the piecewise linear approximation obtained by interpolating the trajectory of Σ_{FEM} . In the following, we assume the partition $\{x_i\}_{i=1}^m$ is proposition preserving with respect to the set of regions $\{X_i\}_{i=1}^m$ ². Suppose we are given an a priori bound,

¹In what follows we use Σ to refer to $\Sigma(u_0, g_d, g_n, v_D, v_N)$.

²I.e., the X_i sets are unions of the regions defined by the partition. This is well defined since the

$\epsilon_i(x, t)$, for the pointwise difference between the i th derivative of the trajectory and the FEM approximation, i.e., $|\frac{\partial^i}{\partial x^i} u(x, t) - \frac{\partial^i}{\partial x^i} u^d(x, t)| \leq \epsilon^i(x, t)$.

Definition 2.1. Let $\lambda \in \Lambda$ be a predicate and let $\delta : \bar{\Omega} \rightarrow \mathbb{R}$ be a continuous and differentiable function. The perturbation of λ by δ , λ^δ , is the predicate given by the tuple $(Q_\lambda, X_\lambda, \mu_\lambda^\delta, D_\lambda)$, where $\mu_\lambda^\delta(x, u) = \mu_\lambda(x, u) + \delta(x)$.

Definition 2.2. Let ψ be an S-STL formula over Λ in negation normal form and let $\delta = \{\delta_i\}_{i=0,1,\dots}$ be a set of continuous functions $\delta_i : \bar{\Omega} \rightarrow \mathbb{R}$. The conservative correction of ψ by δ , ψ^δ , is an S-STL formula in negation normal form obtained by substituting every predicate λ that appears in ψ in the following way:

- If λ is not preceded by a negation operator and $D_\lambda = \frac{d^i}{dx^i}$, then it is substituted by λ^{δ_i} .
- Otherwise, it is substituted by $\lambda^{-\delta_i}$.

We will also use Λ^δ to refer to the set of all δ and $-\delta$ corrections of predicates in Λ .

Theorem 2.2. If $u^d \models \psi^\delta$, with $\delta_i(x) = \max_t \epsilon_i(x, t)$, then $u \models \psi$.

Proof. We only need to consider a predicate λ and its negation. Let i be the order of the derivative D_λ . We have

$$\begin{aligned} \left| \frac{\partial^i}{\partial x^i} u(x, t) - \mu_\lambda(x) - \frac{\partial^i}{\partial x^i} u^d(x, t) + \mu_\lambda(x) \right| = \\ \left| \frac{\partial^i}{\partial x^i} u(x, t) - \frac{\partial^i}{\partial x^i} u^d(x, t) \right| \leq \delta_i(x). \end{aligned} \quad (2.8)$$

Thus, $\frac{\partial^i}{\partial x^i} u(x, t) - \mu_\lambda(x) \geq \frac{\partial^i}{\partial x^i} u^d(x, t) - \mu_\lambda(x) - \delta_i(x)$, which proves the result for $\psi = \lambda$. For $\psi = \neg \lambda$, note that satisfaction is equivalent to $\frac{\partial^i}{\partial x^i} u(x, t) - \mu_\lambda(x) < 0$ for x quantified opposite to Q_λ . Finally, from (2.8) we have $\frac{\partial^i}{\partial x^i} u(x, t) - \mu_\lambda(x) \leq \frac{\partial^i}{\partial x^i} u^d(x, t) - \mu_\lambda(x) + \delta_i(x)$. \square

Example 2.2. Assume we obtain a bound $\delta_0(x) = 0.25$ for the system Σ^H described in Ex. 2.1 and its FEM approximation. Consider the same specification ϕ_{ex} defined

X_i sets are closed and there are a finite number of them. In higher dimensions, we also require the X_i to be polytopes.

in the example. The perturbed specification, ϕ_{ex}^δ is then:

$$\begin{aligned} \phi_{ex}^\delta = & \mathbf{G}_{[4,5]} (\\ & (\forall x \in [30, 60] : u(x) - (\frac{x}{4} + 303 - 0.25) < 0) \wedge \\ & (\forall x \in [30, 60] : u(x) - (\frac{x}{4} + 297 + 0.25) > 0)) \wedge \\ & \mathbf{G}_{[0,5]} (\forall x \in [100, 100] : u(x) - 345 - 0.25 < 0). \end{aligned} \quad (2.9)$$

If the FEM approximation to Σ^H satisfies ϕ_{ex}^δ , we can conclude Σ^H satisfies ϕ_{ex} .

Theorem 2.2 gives us a way to conservatively solve Problem 2.1 using the approximation obtained from Σ_{FEM} . However, we still need to deal with continuous functions in continuous time. Our next step will be to reformulate the specification ψ^δ into an STL formula that can be checked against the trajectory of Σ_{FEM} , i.e., $d(t)$.

Let $\Lambda_{FEM}^\delta = \{\alpha_{\lambda,e} | \lambda \in \Lambda^\delta, e \in E_\lambda\} \cup \{\beta_{\lambda,j} | \lambda \in \Lambda^\delta, j \in J_\lambda, D_\lambda = id\}$, where $E_\lambda = \{e | [x_e, x_{e+1}] \subseteq X_\lambda\}$, $J_\lambda = \{j | x_j \in X_\lambda, [x_{j-1}, x_j] \not\subseteq X_\lambda, [x_j, x_{j+1}] \not\subseteq X_\lambda\}$, and satisfaction of $\alpha_{\lambda,e}$ and $\beta_{\lambda,j}$ by a continuous-time signal $d : [0, T] \rightarrow \mathbb{R}^n$ is defined in the following way:

$$d[t] \models \alpha_{\lambda,e} \iff D_\lambda u^d(x_e^m, t) - \mu_\lambda(x_e^m) > 0, \quad (2.10)$$

$$d[t] \models \beta_{\lambda,j} \iff d_j(t) - \mu_\lambda(x_j) > 0, \quad (2.11)$$

where $x_e^m = \frac{x_e + x_{e+1}}{2}$ is the midpoint of element e . Note that $D_\lambda u^d(x_e^m, t)$ is a function of $d(t)$. The α predicates simplify a predicate λ so that only a representative value (the midpoint) is checked at each element. The β predicates are needed in those cases where X_λ is a single point. The robustness degree for these predicates is defined as in Sec. 2.1.2. Note that this set of predicates includes the perturbations defined in Def. 2.2. We define perturbations of predicates in Λ_{FEM}^δ by a real number k in a manner analogous to Def. 2.1, and we denote it as α^k .

Definition 2.3. *The STL formula over Λ_{FEM}^δ , $\psi_{FEM}^{\delta,\eta}$, corresponding to an S-STL*

formula in negation normal form, ψ^δ , over Λ^δ , with $\eta = \{\eta_i\}_{i=0,1,\dots}, \eta_i : \bar{\Omega} \rightarrow \mathbb{R}$, is a formula obtained by substituting every predicate λ in ψ^δ by the formula $\bigoplus_{e \in E_\lambda} \gamma_{\lambda,e} \oplus \bigoplus_{j \in J_\lambda} \beta_{\lambda,j}$, where $\oplus = \wedge$ if $Q_i = \forall$ and $\oplus = \vee$ otherwise, and $\gamma_{\lambda,e}$ is defined as the following STL formula:

- If λ is not preceded by a negation operator, then $\gamma_{\lambda,e} = \alpha_{\lambda,e}^{-k_e^\lambda}$.
- Otherwise, $\gamma_{\lambda,e} = \alpha_{\lambda,e}^{k_e^\lambda}$.

In both cases, for $D_\lambda = \frac{d^i}{dx^i}$ and $l_e = x_{e+1} - x_e$,

$$k_e^\lambda = \frac{l_e}{2} \left(\max_{c \in [x_e, x_{e+1}]} |\mu'_\lambda(c)| + \max_{c \in [x_e, x_{e+1}]} \eta_i(c) \right). \quad (2.12)$$

Theorem 2.3. If $d \models \psi_{FEM}^{\delta,\eta}$, with $\eta_i(x) \geq \max_t |\frac{\partial^{i+1} u^d}{\partial x^{i+1}}(x, t)|$, then $u^d \models \psi^\delta$.

Proof. We only need to consider a predicate λ and its negation. We assume $Q_\lambda = \forall$, the other case is similar. Let $X_\lambda = [x_a, x_b]$ and i be the order of the derivative D_λ . First note that satisfying λ is equivalent to satisfying all predicates of the set $\{(Q_\lambda, [x_e, x_{e+1}], \mu_\lambda, D_\lambda) | e \in E_\lambda\}$. For any $e \in E_\lambda$, let $x^m = \frac{x_e + x_{e+1}}{2}$, $h = \frac{x_{e+1} - x_e}{2}$. For $x \in [x_e, x_{e+1}]$ we have:

$$\begin{aligned} & \left| \frac{\partial^i}{\partial x^i} u^d(x, t) - \mu_\lambda(x) - \frac{\partial^i}{\partial x^i} u^d(x^m, t) + \mu_\lambda(x^m) \right| \leq \\ & h \max_{c \in [x_e, x_{e+1}]} \left| \mu'_\lambda(c) + \frac{\partial^{i+1} u^d}{\partial x^{i+1}}(c, t) \right| \leq \\ & h \left(\max_{c \in [x_e, x_{e+1}]} |\mu'_\lambda(c)| + \max_{c \in [x_e, x_{e+1}]} \left| \frac{\partial^{i+1} u^d}{\partial x^{i+1}}(c, t) \right| \right) \leq \\ & h \left(\max_{c \in [x_e, x_{e+1}]} |\mu'_\lambda(c)| + \max_{c \in [x_e, x_{e+1}]} \eta_i(c) \right) = K_e. \end{aligned} \quad (2.13)$$

Then,

$$\begin{aligned} \frac{\partial^i}{\partial x^i} u^d(x, t) - \mu_\lambda(x) & \geq \frac{\partial^i}{\partial x^i} u^d(x^m, t) - \mu_\lambda(x^m) - K_e = \\ & D_\lambda u^d(x^m, t) - \mu_\lambda(x^m) - K_e, \end{aligned} \quad (2.14)$$

so $d \models \alpha_{\lambda,e}^{-K_e}$ implies $u^d \models (Q_\lambda, [x_e, x_{e+1}], \mu_\lambda, D_\lambda)$ and the theorem holds for $\psi^\delta = \lambda$. To prove it for the negated predicate, we can follow an argument similar to the one in the proof of Thm. 2.2. \square

Example 2.3. Continuing with Ex. 2.2, assume we obtained the FEM approximation using the partition $\{10i | i \in 0, \dots, 10\}$ and we found the bound $\eta_0(x) = 0.15$. The perturbed STL specification corresponding to ψ^δ is

$$\begin{aligned} \phi_{ex,FEM}^{\delta,\eta} = \mathbf{G}_{[4,5]}(& \\ & (y_4 - 311.75 - 0.25 - 5 * (0.25 + 0.15) < 0) \wedge \\ & (y_5 - 314.25 - 0.25 - 5 * (0.25 + 0.15) < 0) \wedge \\ & (y_6 - 316.75 - 0.25 - 5 * (0.25 + 0.15) < 0) \wedge \\ & (y_4 - 305.75 - 0.25 - 5 * (0.25 + 0.15) > 0) \wedge \\ & (y_5 - 308.25 - 0.25 - 5 * (0.25 + 0.15) > 0) \wedge \\ & (y_6 - 310.75 - 0.25 - 5 * (0.25 + 0.15) > 0)) \wedge \\ & \mathbf{G}_{[0,5]}(d_{11} - 345 < 0), \end{aligned} \quad (2.15)$$

where $y_e = u^d(x_e^m) = \frac{d_e + d_{e+1}}{2}$. If we prove satisfaction of $\phi_{ex,FEM}^{\delta,\eta}$ by Σ_{FEM}^H , we can conclude the interpolation satisfies ϕ_{ex}^δ .

Finally, we reformulate the specification into an STL formula with discrete time semantics that can be checked against the trajectory of a time discretization of Σ_{FEM} with time interval $\Delta t \in \mathbb{R}_{>0}$. There are several options at this point: the simplest one is to consider Σ_{FEM} as a first order linear system (augmenting the state space if needed) and define the time discretization as the following difference equation:

$$\Sigma_{FEM}^{\Delta t}(d(0), g) \quad \begin{cases} \tilde{d}^{k+1} = \tilde{A}\tilde{d}^k + \tilde{b}(g), \\ \tilde{d}^0 = d(0), \end{cases} \quad (2.16)$$

where $\tilde{A} = e^{A\Delta t}$ and $\tilde{b}(g) = -e^{A\Delta t}A^{-1}(e^{-A\Delta t} - I)b(g)$. Note that, in theory, $d(k\Delta t) = \tilde{d}^k, \forall k \in \mathbb{N}$. However, in practice one needs to numerically compute the exponential matrices in \tilde{A} and \tilde{b} , which introduces an approximation error difficult to control. As an alternative, we can use any numerical integration algorithm with fixed time step appropriate to the specific PDE system under study, several of which have been thoroughly analyzed in the FEM literature, such as the Newmark family. Similar to

the FEM approximation, assume we have a bound on the approximation error of the integration algorithm, $\max_k |d_j(k\Delta t) - \tilde{d}_j^k| \leq \epsilon_j^d, j = 1, 2, \dots$

We define a conservative correction of $\psi_{FEM}^{\delta, \eta}$ by a real vector $\nu = (\nu^y, \nu^d)$, $\psi_{FEM}^{\delta, \eta, \nu}$, in a similar way to Def. 2.3, noting that, for $D_\lambda = \frac{\partial^i}{\partial x^i}$, the predicate $\gamma_{\lambda, e}$ is perturbed using the constant $\nu_{i, e}^y$ and the predicate $\beta_{\lambda, j}$ is perturbed using the constant ν_j^d . We abuse the notation for STL formulas so that we can consider satisfaction of the discrete time signal \tilde{d} . In particular, $\tilde{d}[t] \models \mu \iff \mu(\tilde{d}^{\lfloor t/\Delta t \rfloor}) > 0$.

Theorem 2.4. *If $\tilde{d} \models \psi_{FEM}^{\delta, \eta, \nu}$, with $\nu_j^d \geq \Delta t \max_t |\dot{d}_j(t)| + \epsilon_j^d$ and $\nu_{i, e}^y \geq \Delta t \max_t |\frac{d}{dt} D^i u^d(x_e^m, t)| + D^i u^{\epsilon^d}(x_e^m)$, then $d \models \psi_{FEM}^{\delta, \eta}$.*

Proof. Again we only need to consider a predicate and its negation. We will assume the predicate is of the form $\gamma_{\lambda, e}$, as the predicate $\beta_{\lambda, j}$ is a particular case. Let i be the order of the derivative in the predicate, $D_\lambda = \frac{\partial^i}{\partial x^i}$, and let $k = \lfloor t/\Delta t \rfloor$. First note that the bound on the integration error ϵ_j^d gives the following bound:

$$\begin{aligned} |D^i u^d(x_e^m, k) - D^i u^{\tilde{d}}(x_e^m)| &\leq D^i u^{\epsilon^d}(x_e^m) \implies \\ D^i u^d(x_e^m, k) &\geq D^i u^{\tilde{d}}(x_e^m) - D^i u^{\epsilon^d}(x_e^m). \end{aligned} \quad (2.17)$$

Then, similar to the proof of Thm. 2.3, we have the following:

$$\begin{aligned} |D^i u^d(x_e^m, t) - \gamma_{\lambda, e} - D^i u^d(x_e^m, k) + \gamma_{\lambda, e}| &= \\ |D^i u^d(x_e^m, t) - D^i u^d(x_e^m, k)| &\leq \\ \Delta t \max_{t \in [k, k+1]} \left| \frac{d}{dt} D^i u^d(x_e^m, t) \right| &\leq \\ \Delta t \max_t \left| \frac{d}{dt} D^i u^d(x_e^m, t) \right| &= M_e. \end{aligned} \quad (2.18)$$

Combining the bounds obtained in (2.17) and (2.18) we have:

$$\begin{aligned} D^i u^d(x_e^m, t) - \gamma_{\lambda, e} &\geq D^i u^d(x_e^m, k) - \gamma_{\lambda, e} - M_e \geq \\ &D^i u^{\tilde{d}}(x_e^m) - \gamma_{\lambda, e} - M_e - D^i u^{\epsilon^d}(x_e^m), \end{aligned} \quad (2.19)$$

which proves the theorem for the positive predicate. To prove it for the negated predicate, take the opposite bound for the absolute values in (2.17) and (2.18) and combine them in a similar way to (2.19). \square

Example 2.4. *Continuing with Ex. 2.3, assume we discretize $\Sigma_{FEM}^H(u_0, g)$ with perfect accuracy using the timestep $\Delta t = 0.005$ and let $\nu = (0.5, 0.5, \dots)$. The final corrected specification is*

$$\begin{aligned} \phi_{ex, FEM}^{\delta, \eta, \nu} = \mathbf{G}_{[4,5]}(& \\ & (y_4 - 311.75 - 0.25 - 5 * (0.25 + 0.15) + 0.5 * 0.005 < 0) \wedge \\ & (y_5 - 314.25 - 0.25 - 5 * (0.25 + 0.15) + 0.5 * 0.005 < 0) \wedge \\ & (y_6 - 316.75 - 0.25 - 5 * (0.25 + 0.15) + 0.5 * 0.005 < 0) \wedge \\ & (y_4 - 305.75 - 0.25 - 5 * (0.25 + 0.15) + 0.5 * 0.005 > 0) \wedge \\ & (y_5 - 308.25 - 0.25 - 5 * (0.25 + 0.15) + 0.5 * 0.005 > 0) \wedge \\ & (y_6 - 310.75 - 0.25 - 5 * (0.25 + 0.15) + 0.5 * 0.005 > 0)) \wedge \\ & \mathbf{G}_{[0,5]}(d_{11} - 345 + 0.5 * 0.005 < 0), \quad (2.20) \end{aligned}$$

We can guarantee the satisfaction of $\phi_{ex, FEM}^{\delta, \eta}$ by Σ_{FEM}^H if $\Sigma_{FEM}^{H, \Delta t}$ satisfies $\phi_{ex, FEM}^{\delta, \eta, \nu}$.

The main result in this work is a corollary previous theorems, which allows us to solve Problem 2.1 by solving a control problem for discrete-time linear systems with regular STL constraints:

Theorem 2.5. *If $\Sigma_{FEM}^{\Delta t} \models \psi_{FEM}^{\delta, \eta, \nu}$, with δ, η and ν defined as in Thms. 2.2 to 2.4 and $d_i^0 = u_0(x_i), i = 1, \dots, n$, with u_0 an initial value for Σ , then $\Sigma \models \psi$.*

Proof. From Thm. 2.4, if $\Sigma_{FEM}^{\Delta t} \models \psi_{FEM}^{\delta, \eta, \nu}$, which is shorthand for $\tilde{d} \models \psi_{FEM}^{\delta, \eta, \nu}$, then $d \models \psi_{FEM}^{\delta, \eta}$, with d the trajectory of Σ_{FEM} . From Thm. 2.3, we now have $u^d \models \psi^\delta$. Finally, from Thm. 2.2, we have $u \models \psi$, which means $\Sigma \models \psi$. \square

We can also obtain a bound for the robustness of the trajectory of Σ with respect to the original specification by making the following observation:

Theorem 2.6. *If u, u^d, d and \tilde{d} are trajectories of Σ , interpolation of Σ_{FEM} , Σ_{FEM} and $\Sigma_{FEM}^{\Delta t}$, respectively, and ψ is an S-STL formula over Λ , then the following inequality holds:*

$$r(\psi, u, t) \geq r(\psi^\delta, u^d, t) \geq r(\psi_{FEM}^{\delta, \eta}, d, t) \geq r(\psi_{FEM}^{\delta, \eta, \nu}, \tilde{d}, t). \quad (2.21)$$

Proof. Again, we only need to prove it for a predicate. Consider the following inequalities in the proofs of Thms. 2.2 to 2.4:

$$\begin{aligned}
\frac{\partial^i}{\partial x^i} u(x, t) - \mu_\lambda(x) &\geq \frac{\partial^i}{\partial x^i} u^d(x, t) - \mu_\lambda(x) - \delta_i(x) = \\
\frac{\partial^i}{\partial x^i} u^d(x, t) - \mu_\lambda(x) - \delta_i(x) &\geq D_\lambda u^d(x^m, t) - \mu_\lambda(x^m) - \delta_i(x) - K_e = \\
D^i u^d(x_e^m, t) - \gamma_{\lambda, e} &\geq D^i u^{\tilde{d}}(x_e^m) - \gamma_{\lambda, e} - M_e - D^i u^{\epsilon^d}(x_e^m).
\end{aligned} \tag{2.22}$$

Note that we have added the first correction term explicitly in the second inequality, as predicates at that point include them implicitly. Each of the terms in these inequalities is equal to the robustness of the corresponding formula with respect to the trajectory, which proves the theorem for a positive predicate. For the negative case, note that $r(-\lambda, u, t) = -r(\lambda, u, t)$ and that the inequalities above are flipped in the case of negative predicates with the appropriate sign changes. \square

2.5 MILP Formulation of Control Synthesis

In this section, we solve Problem 2.1 by formulating an optimization problem using the corrected STL specification defined in the previous section. Our formulation is equivalent to an MILP, which we solve using an off-the-shelf solver such as Gurobi (Gurobi Optimization, 2016). The optimization problem takes the following form:

$$\begin{aligned}
r_m = \max \quad & r(\psi_{FEM}^{\delta, \eta, \nu}, \tilde{d}, 0) \\
\text{s.t.} \quad & \tilde{d}^{k+1} = \tilde{A}\tilde{d}^k + \tilde{b}(g), \\
& \tilde{d}^0 = d(0), \\
& v_D \in V_D, v_N \in V_N.
\end{aligned} \tag{2.23}$$

In the above, (2.16) should be substituted by the appropriate difference equations resulting from the chosen ODE integration algorithm. After solving (2.23), if $r_m > 0$, then ψ is satisfied by the controlled system using as control inputs the optimal solution for v_D and v_N . This optimization problem is clearly non-convex, due to the max and

min operators present in the definition of robustness, and the objective function is non-differentiable. However, we can apply the technique described in (Sadraddini and Belta, 2015) to represent robustness as mixed-integer linear constraints. On the other hand, the system dynamics are linear and our assumption on the shape of the admissible control sets V_D and V_N implies that they can be encoded as linear constraints. Therefore, (2.23) is a MILP. Also note that by using the robustness degree as cost function, (2.23) is always feasible and a control input will be produced even if it does not correspond to a satisfying trajectory. In this case, r_m would be negative and we can think of the resulting optimal values for v_D and v_N as best effort inputs.

The computational time needed to solve a MILP grows exponentially with the number of binary variables. In our encoding, we introduce one binary variable for each argument to a min or max function. Thus, the number of binary variables is proportional to the length of time intervals (in the discrete sense), the number of boolean connectives in the original formula ψ and the length of the discretized predicates obtained in Def. 2.3. In terms of parameters of the problem and solution, the length of the discrete-time intervals is proportional to the length of the time intervals in ψ and inversely proportional to Δt ; regarding the spatial discretization of the predicates, its length is proportional to the size of the spatial domains and inversely proportional to the size of the partition (i.e., the distance between two nodes in the partition).

Note that in the optimization problem (2.23), the only decision variables present are the ones encoding the control inputs. Given our assumptions, these variables will almost always be real-valued and constrained to an interval. In the MILP formulation, all other decision variables are introduced to encode the trajectory of the system and the robustness with respect to the specification. Crucially, all integer decision

variables are only needed to encode the robustness, and their values are completely determined once a control input is selected. Given these observations, we can improve the performance of the MILP solver by precomputing an approximation to the optimal control inputs and use it as a starting point for the MILP solver. This approximation can be computed using a gradient-free optimization algorithm such as differential evolution (Storn and Price, 1997). From this approximation, the values of the MILP decision variables can be inferred and used as a feasible starting solution, which greatly improves the performance of the solver by allowing it to cut solutions with worse robustness.

2.6 Extension to Multi-Dimensional PDEs

In this section we sketch how to extend our control synthesis framework to vector fields in multi-dimensional domains, with a slight generalization of the spatial predicates to allow linear functions of the state variables. The first step is to define an appropriate extension of S-STL to handle the extra state dimensions and extra directional derivatives. Then, the specification is discretized following the same steps defined in Sec. 2.4, using slightly modified correction terms. The PDE discretization using the FEM follows the same principles as in the one-dimensional case, and we refer the reader to (Hughes, 2000) for more details. Finally, the MILP has the same formulation described in Sec. 2.5. A detail worth noting is the control inputs are now potentially functions of space if we are to design a boundary control input that is applied to a region in the boundary. The same assumption on the shape of the admissible control sets as in one-dimensional problems is made. In what follows, we assume $u(x, t) : \bar{\Omega} \times [0, T] \rightarrow \mathbb{R}^m$ is a vector field on $\Omega \subset \mathbb{R}^n$, with u_j its j th component.

Let $\mathcal{D} = \{id, \frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_n}\}$ be the set of partial spatial derivatives plus the identity

operator, and let \mathcal{D}^* be the monoid over \mathcal{D} with composition of operators. Finally, let D^j , with $D \in \mathcal{D}^*$ and $j = 1, \dots, m$, be the operator defined as $(D^j u)(x, t) = (Du_j)(x, t)$. We define predicates in multi-dimensional S-STL as the following:

$$\lambda \equiv Qx \in X : \sum_{i=1}^k a_i (D_i^{j_i} u)(x) - \mu(x) > 0, \quad (2.24)$$

where Q, X and μ are defined as in Sec. 2.2, $a = (a_i)_{i=1}^k \in \mathbb{R}^k$ is a vector of coefficients and $D_i \in \mathcal{D}^*$. Note that, besides supporting a vector field u on a multi-dimensional domain, the extended predicates allow linear functions of u and its derivatives. Multi-dimensional S-STL formulas are defined over a set of extended predicates as described in Sec. 2.2.

Discretization of multi-dimensional S-STL follows the same steps as the one-dimensional case. However, the correction terms must be redefined. In order to simplify the notation, we consider only one predicate λ defined as in (2.24).

- δ corrections in Thm. 2.2. Assume we have approximation error bounds $|(D_i^{j_i} u)(x, t) - (D_i^{j_i} u^d)(x, t)| \leq \epsilon_{D_i^{j_i}}(t)$ for all $D_i^{j_i}$ that appear in λ . The multi-dimensional δ perturbation of a predicate must be defined with $\mu^\delta = \mu(x) + \|a\| \sqrt{\sum_{i=1}^k (\delta_{D_i^{j_i}})^2}$, and $\delta_{D_i^{j_i}} = \max_t \epsilon_{D_i^{j_i}}(t)$.
- η corrections in Thm. 2.3. The discretization scheme for predicates is identical if we define as element midpoints, x_e^m , the Chebyshev center of the element. Let h_e be the Chebyshev radius of element e , and $X_e \subset \bar{\Omega}$ the region that defines element e . The multi-dimensional η perturbation is defined now with $k_e = h_e (\max_{c \in X_e} \|\nabla \mu(c)\| + \sqrt{\eta} \|a\| \max_{c \in X_e} \sqrt{\sum_{i=1}^k \sum_{l=1}^n (\eta_{\frac{\partial}{\partial x_l} D_i^{j_i}}(c))^2})$, with η_{D^j} being bounds on the partial derivatives of the FEM approximation, $\eta_{D^j} \geq \max_t |(D^j u^d)(x, t)|$.
- ν corrections in Thm. 2.4. In this case, we modify the perturbation constants to

$l_e^y = \Delta t \|a\| \sqrt{\sum_{i=1}^k (\nu_{e,D_i^j}^y)^2}$, with $\nu_{e,D_i^j}^y \geq \max_t | \frac{d}{dt} (D_i^{j_i} u^d)(x_e^m, t) | + (D_i^{j_i} u^{\epsilon^d})(x_e^m)$ and similarly for the l_j^d constant.

2.7 Extension to Non-Linear Equations

Much of the interesting physical phenomena seen in materials involve both material and geometric nonlinearities. In this section, we discuss how the proposed approaches can be adapted to deal with material nonlinearities. We discuss these phenomena in the context of an elastodynamics problem (wave equation) for ease of exposition, although similar concepts apply to other PDEs.

Material nonlinearities introduce nonlinear terms in the internal force term of the PDE (e.g., the second term in first line of Equation (2.1)) via specific constitutive equations. For example, for an elastodynamics problem using rubbery elastomeric materials the material could be represented using the nonlinear Gent constitutive law, instead of the usual Hooke's law for linear elasticity. While the kinematic description changes in the presence of these nonlinearities, what is essential is that the form of the ODE that results from the FEM discretization of the PDE is the same. In the case of the nonlinear wave equation, the FEM-discretized ODE retains the form:

$$\Sigma_{FEM}^{NL} : \quad M \ddot{d} = f^{ext} - f^{int}(d), \quad (2.25)$$

where d are the displacements. In the linear case, the internal force is $f^{int} = Kd$, with K being the (constant) stiffness matrix. In the nonlinear case, the internal force $f^{int}(d)$ is a nonlinear function of the displacements d (for example, $f^{int}(d) = K(d)d$).

In our approach, the linearity assumption on the resulting ODE is a technical requirement: the MILP encoding relies on this assumption. We propose to solve the problem in the nonlinear case by approximating the system with a hybrid system in the following way: suppose that we can define a polyhedral partition of the state space

of the ODE system, $\{D_i\}_{i \in I}$, such that the system has a good linear approximation at each region. We define a hybrid linear approximation of Σ_{FEM}^{NL} as:

$$\Sigma_{FEM}^L : \quad M\ddot{d} = f^{ext} - K_i d, \quad d \in D_i, i \in I, \quad (2.26)$$

where K_i is related to the differential of f^{int} inside D_i .

Hybrid systems of this form admit MILP encodings (Bemporad and Morari, 1999). However, the complexity of the optimization problem can explode very quickly even for very simple examples if the encoding technique is applied naively. Consider for example a simple nonlinearity in the wave equation where Hooke's law $\sigma = E\epsilon$, with σ being the stress, E the Young's modulus and ϵ the strain, becomes the piecewise linear law:

$$\begin{aligned} \sigma &= E_1 \epsilon, \epsilon \leq \epsilon_y, \\ \sigma &= E_2 \epsilon, \epsilon > \epsilon_y, \end{aligned} \quad (2.27)$$

with ϵ_y the yield strain of the material and E_1, E_2 constants. When translating this to the FEM system, we can say that each element is in one of two linear modes described by one of the equations above. Thus, the full hybrid system is described by $2^{\text{number of elements}}$ modes. The MILP encoding for this system would use a number of binary variables equals to the number of modes times the number of time steps, which is a prohibitively large number. Instead of this naive approach, we propose an implicit representation of the modes of the hybrid system, which reduces the necessary number of binary variables per time step to just the number of elements.

In order to formulate this implicit representation, note that the stiffness matrix, $K(d)$, resulting from the FEM is constructed as follows:

$$K(d) = \sum_{j=1}^m K_j(d), \quad (2.28)$$

where K_j is the local stiffness matrix to element j and depends on the mode of

the element as given by (2.27) when considered locally. The full ODE system then becomes:

$$M\ddot{d} = f^{ext} - \sum_{j=1}^m K_j(d)d, \quad (2.29)$$

where $K_j(d)$ is either of the constant matrices K_j^1 or K_j^2 depending only on the values of d at element j . This choice can be encoded in the MILP using only one binary variable per element per time step, greatly improving the performance over explicitly encoding each mode of the full system.

2.8 Verification for Sets of Initial Values and Boundary Conditions

As a byproduct of our control synthesis framework, we also obtain a method for the verification of S-STL properties in PDE systems. Let us consider that v_D and v_N represent unknown boundary conditions instead of control inputs. A verification problem can be formulated as follows:

Problem 2.2 (Set of boundary conditions verification). *Given a PDE Σ , an initial value u_0 , prescribed boundary conditions g_d and g_n , an S-STL formula ψ over a set of predicates Λ , and unknown boundary conditions v_D and v_N constrained to be within the sets V_D and V_N , respectively, check whether the trajectory of $\Sigma(u_0, g_d, g_n, v_D, v_N)$ satisfies ψ for all $v_D \in V_D$ and $v_N \in V_N$.*

This problem can be solved by using a MILP formulation similar to (2.23). However, in this case the optimization objective is to minimize the robustness. Then, if $r_m > 0$ we guarantee that ψ is satisfied for all possible boundary conditions. A similar procedure can be followed if we are interested in verifying that a specification is met for all initial values of the PDE within a set.

2.9 Case Study

In this section we solve Problem 2.1 for the heat equation example introduced earlier in the paper. We also discuss the conservativeness of our approach as well as the computational performance. We implemented our framework using Python 2.7, Gurobi 7.0 as our MILP solver configured to use 10 threads and the differential evolution implementation in scipy 0.17.1 with the default parameters and capped to a maximum of 50 iterations. We ran our implementation in an Intel Core i7-5930K and 16GB RAM.

2.9.1 Heat Equation

We consider the specification described in Ex. 2.1. We assume the rod is made of two different materials: the section from 30 to 60 mm is made of a material with parameters $E_a = 1500 \cdot 10^3$, $\rho_a = 4.5 \cdot 10^{-6}$ and $c_a = 0.38 \cdot 10^9$, while the rest of the rod is made of a material with parameters $E_b = 800 \cdot 10^3$, $\rho_b = 4 \cdot 10^{-6}$ and $c_b = 0.466 \cdot 10^9$. The applied heat source appears in (2.5) as an additional force in the right hand side, $f_{nodal}(t) = (0, \dots, 0, U(t))^T$, with $U(t)$ constrained to be continuous, piecewise linear and $U(t) \in [0, 10^6], \forall t$. We predefine the interpolation times for $U(t)$ to $\{0.5i | i = 0, 1, \dots\}$. The rod starts at temperature 300 K at all points. We partition the spatial domain into a uniform mesh of different sizes and integrate the resulting FEM system using the trapezoidal rule with $\Delta t = 0.05$. For simplicity, we do not consider the integration error, and we approximate the ϵ, η, ν bounds by taking 100 samples of system trajectories with randomized control inputs, obtaining from them approximate maximum spatial and time derivatives, and also the maximum approximation error when considering the true system trajectory the one obtained from an FEM model with 200 elements and $\Delta t = 0.005$. This process is automatic and takes less than 20 seconds.

We used a 30 element mesh to synthesize a control input from the specification ϕ_{ex} , which produces a trajectory with robustness $r(\phi_{ex}) \geq 0.65$. The control input and snapshots of the temperature evolution are shown in Fig. 2·2, along with the temperature profiles considered in ϕ_{ex} (labeled A, B and C in the order they appear in the formula) and a visualization of the total correction applied during the discretization steps. It took 6 seconds to solve the resulting MILP with 13741 variables, 664 of them binary, and 14386 constraints. Furthermore, we show in Fig. 2·2d the relationship between the number of elements of the FEM partition and the conservativeness and computational complexity of the method. Note that for a 10 element mesh, the method fails to obtain a (provably) satisfying control input, and as quality of the mesh improves, so does the bound on the robustness.

2.9.2 Verification Problem For Heat Equation

Continuing with the heat equation example, we now demonstrate how our method can be used for verification. First, we consider the inputs obtained in the previous section for the specification ϕ_{ex} using a 30 element mesh and verify whether the specification is satisfied for different mesh sizes. We show the robustness computed using the specification with and without the corrections in Fig. 2·3a. Note how only using a 20, 30, 40 or 50 element mesh we would verify the system as satisfying the specification. Even though finer meshes are more precise, the inputs generated with a coarser mesh do not necessarily produce a satisfying result when verified with a finer mesh. However, Thm. 2.5 guarantees that the real system satisfies the specification, as can be seen by the (close to) true robustness computed using a fine mesh and no corrections.

We can also verify the specification against unknown but bounded conditions. For example, using the previous set up, consider now that the inputs cannot be precisely applied to the system but within some known tolerance. We show in Fig. 2·3b a

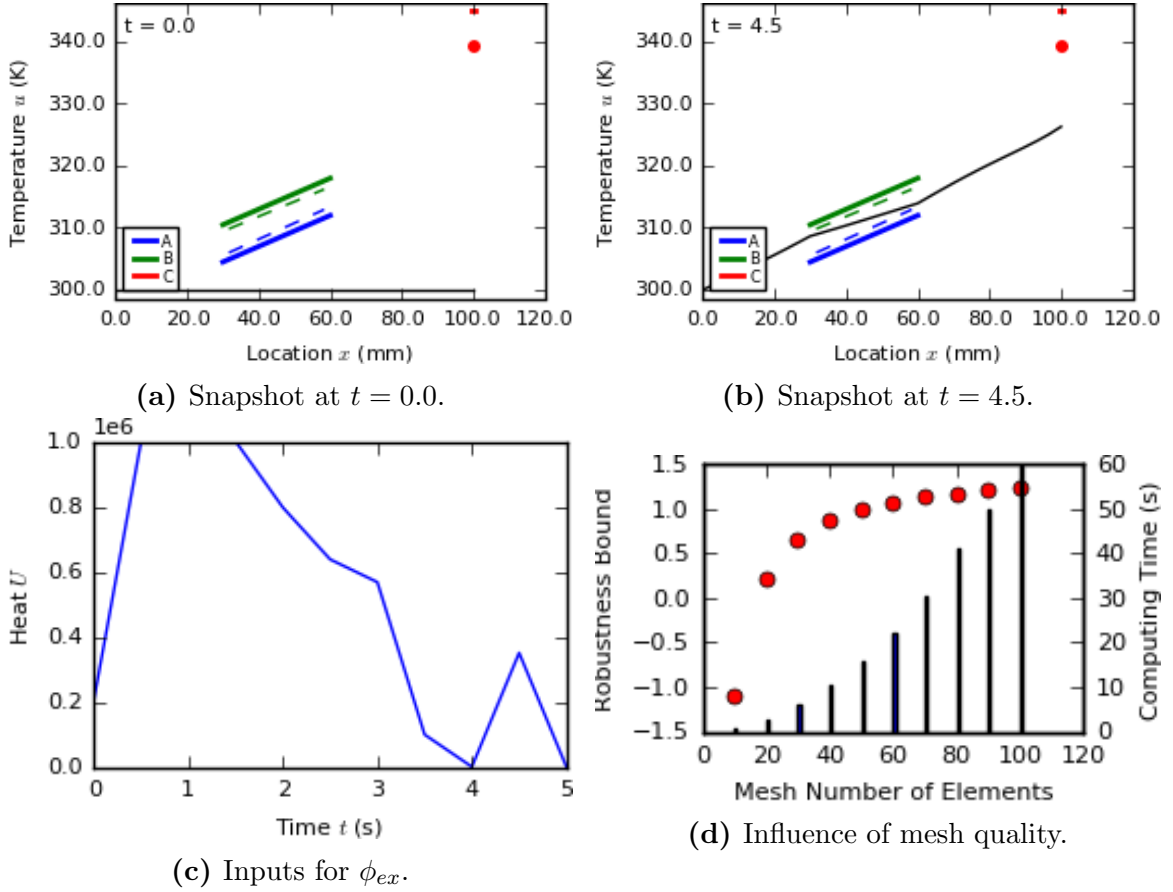


Figure 2.2: In Figs. 2.2a and 2.2b we show snapshots of the satisfying trajectory for ϕ_{ex} in black, with predicate profiles and their corrections in solid and dashed lines respectively, except for the C profile, which is shown as a single dash and dot. In Fig. 2.2c we show the synthesized input. In Fig. 2.2d, we show the computing time (in black bars) and lower bound of the robustness (in red dots) as we increase the number of elements in the mesh.

lower bound on the robustness of the specification for different tolerances using a 30 element mesh. Again, a positive robustness bound signifies satisfaction. Solving the verification problem took around 30 seconds for each one.

2.9.3 Elastic Wave Propagation

Consider a steel and brass rod of length $L = 100$ m, the section between 30 m and 60 m being brass, with densities $\rho_{st} = 8 \cdot 10^3$ kg/m³, $\rho_{br} = 8.5 \cdot 10^3$ kg/m³ and Young's

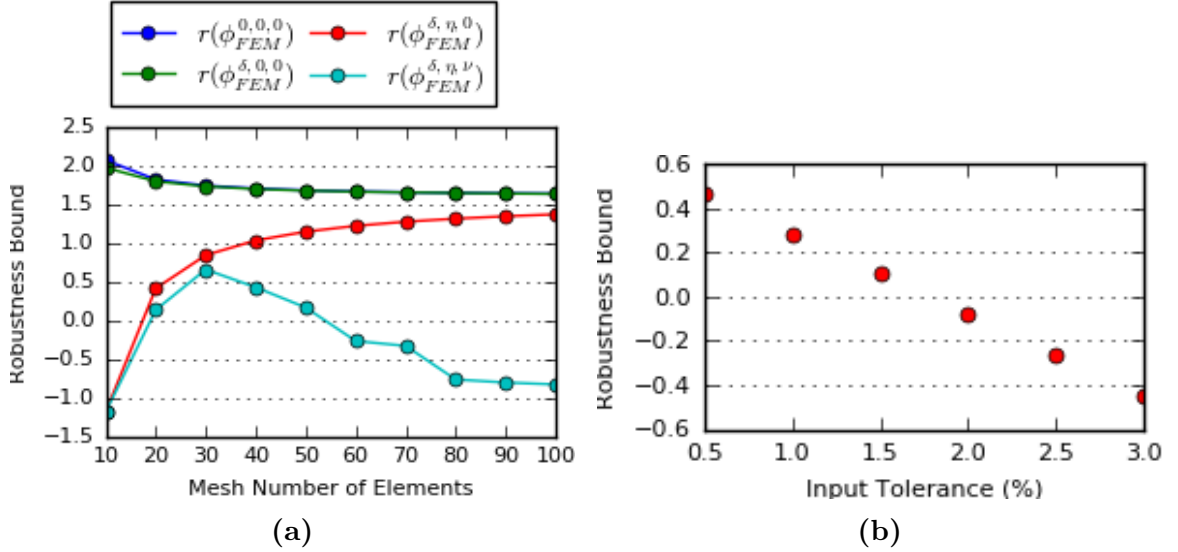


Figure 2.3: Verification results. On the left, robustness lower bounds for a mesh of different size using fixed inputs obtained with a 30 element mesh. On the right, robustness bound while allowing some tolerance over the inputs.

modulus $E_{st} = 200$ GPa, $E_{br} = 100$ GPa, with one end fixed and a time-variant force $U(t)$ applied to the other end. Assume the rod is initially at rest. The displacement $u(x, t)$ of the rod obeys the following IBVP:

$$\Sigma^M(0, U) \left\{ \begin{array}{l} \rho \frac{\partial^2 u}{\partial t^2} - E \frac{\partial^2 u}{\partial x^2} = 0, \text{ on } \Omega \times (0, T), \\ u(0, t) = 0, \forall t \in (0, T), \\ E \frac{\partial u}{\partial x}(L, t) = U(t), \forall t \in (0, T), \\ u(x, 0) = 0, \forall x \in \Omega, \\ \frac{\partial u}{\partial t}(x, 0) = 0, \forall x \in \Omega. \end{array} \right. \quad (2.30)$$

Note that in this mixed material case, $\rho = \rho(x)$ and $E = E(x)$ are the density and Young's modulus of the rod at each point. We build an FEM approximation using a uniform partition with 20 elements and integrate the resulting second order system using the trapezoidal rule (Hughes, 2000) to obtain a time discretization with time

interval $\Delta t = 2.5 \cdot 10^{-3}$ s. We obtain approximate bounds for ϵ, η, ν as in Sec. 2.9.1.

We formulate a control synthesis problem where $U(t)$ is a Neumann control input constrained to be continuous, piecewise affine and $U(t) \in [-5000 \text{ N}, 5000 \text{ N}], \forall t$. Throughout this section, we predefine the interpolation times for $U(t)$ to $\{0.1i | i = 0, 1, \dots\}$. First, we formulate a specification where we require that the rod must be stretched over a given profile for a period of time, then compressed for at least one instant in a given interval, then a choice is given between holding the compressed pattern or returning to the stretched one, and finally the stretched pattern must be achieved within a time interval. Requirements where a material must be stretched or compressed following a profile are important in manufacturing processes such as forming. The resulting S-STL formula is the following:

$$\begin{aligned}
\phi_1 = & \mathbf{G}_{[0.1,0.3]}(\forall x \in [60, 90] : u(x) - \mu_B > 0) \wedge \\
& \mathbf{F}_{[0.3,0.4]}(\forall x \in [60, 90] : u(x) - \mu_C < 0) \wedge \\
& (\mathbf{G}_{[0.45,0.5]}(\forall x \in [60, 90] : u(x) - \mu_C < 0) \vee \\
& \mathbf{G}_{[0.45,0.5]}(\forall x \in [60, 90] : u(x) - \mu_B > 0)) \wedge \\
& \mathbf{F}_{[0.5,0.55]}(\forall x \in [60, 90] : u(x) - \mu_B > 0),
\end{aligned} \tag{2.31}$$

where $\mu_B = 0.005x \cdot 10^{-3} + 0.3$ and $\mu_C = 0$. The specific form of the target profiles was selected considering the feasible shape changes the rod can achieve. The resulting $U(t)$ is shown in Fig. 2·4b, which corresponds to a trajectory with a robustness degree of at least 1.226. We show a snapshot of the trajectory in Fig. 2·4a. It took 165 seconds to solve the problem using a differential evolution presolving step and a MILP with 29345 variables, 1490 of them binary, and 32291 constraints.

We can also formulate a specification where a point in an interval of the rod must be over a given profile for some time while the rod stays below a safe displacement throughout another interval. An S-STL formula describing this specification is:

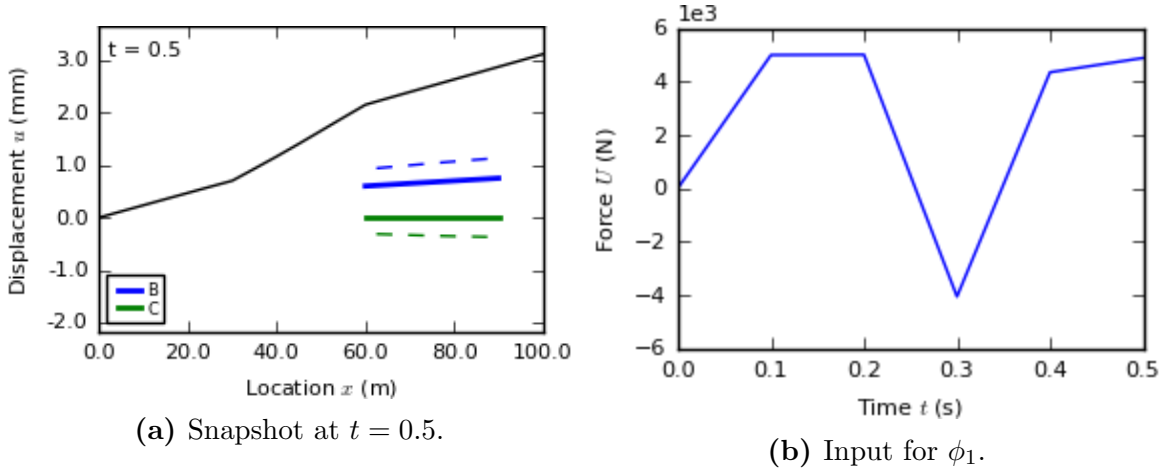


Figure 2.4: Snapshot of the satisfying trajectory for ϕ_1 and synthesized control inputs. Trajectory is shown in black, predicate profiles in solid colored lines and corresponding corrected profiles in dashed lines.

$$\begin{aligned} \phi_3 = & \mathbf{G}_{[0.2,0.3]}(\exists x \in [50, 70] : u(x) - \mu_B > 0) \wedge \\ & \mathbf{G}_{[0.0,0.3]}(\forall x \in [60, 90] : u(x) - \mu_C < 0), \end{aligned} \quad (2.32)$$

where $\mu_B = 0.02x \cdot 10^{-3}$ and $\mu_C = 2.75$. The resulting inputs and a snapshot of the trajectory is shown in Fig. 2.5. The trajectory has robustness degree of at least 0.037 and it took 301 seconds to solve the problem.

Finally, we consider a specification involving the strain, $\frac{d}{dx}u$, of the system, where the objective is to keep the strain in the brass section under a safe profile for some time, and then increase it so that the material yields or breaks within a time window. For this example we predefine $U(0) = 0$. The corresponding S-STL formula is the following:

$$\begin{aligned} \phi_2 = & \mathbf{G}_{[0.1,0.4]}(\forall x \in [30, 60] : \frac{d}{dx}u(x) - \mu_A < 0) \wedge \\ & \mathbf{F}_{[0.4,0.5]}(\forall x \in [30, 60] : \frac{d}{dx}u(x) - \mu_C > 0), \end{aligned} \quad (2.33)$$

where $\mu_A = 2 \cdot 10^{-5}$, $\mu_C = 3 \cdot 10^{-5}$. The control input obtained is shown in Fig. 2.6b. It corresponds to a trajectory with $8.51 \cdot 10^{-6}$ robustness bound and it was synthesized in 123 seconds. We show a snapshot of the trajectory in Fig. 2.6a.

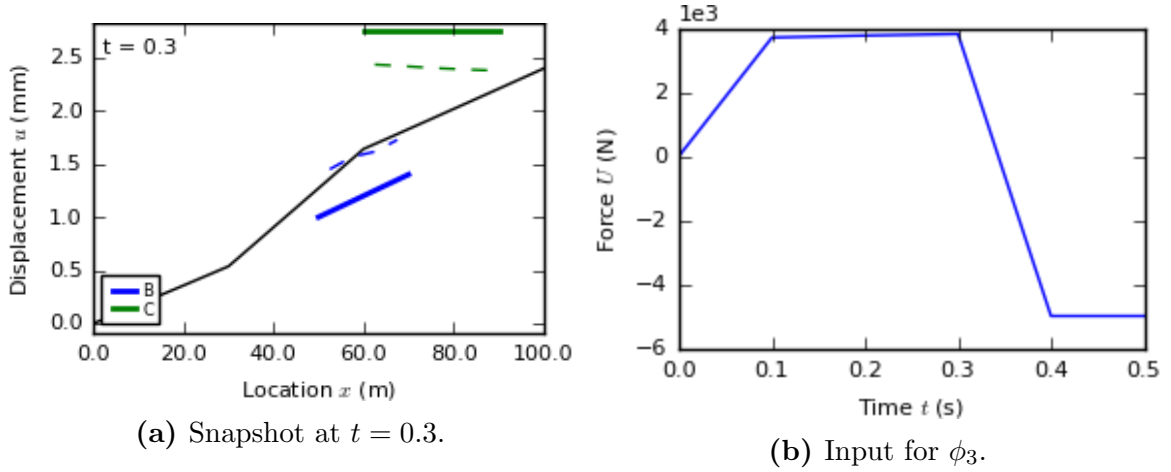


Figure 2-5: Snapshot of the satisfying trajectory for ϕ_3 and synthesized control inputs. Trajectory is shown in black, predicate profiles in solid colored lines and corresponding corrected profiles in dashed lines.

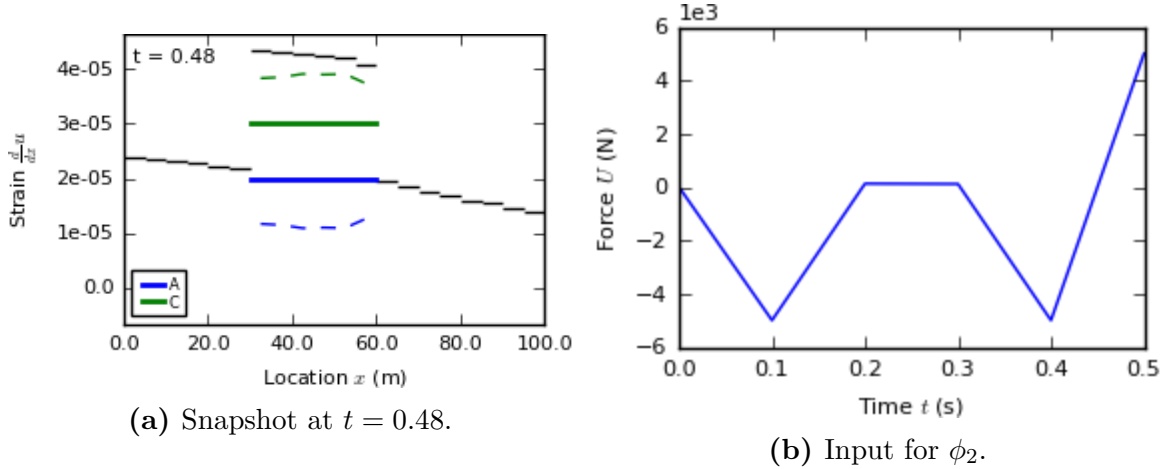


Figure 2-6: Snapshot of the satisfying trajectory for ϕ_2 and synthesized control inputs. Trajectory is shown in black, predicate profiles in solid colored lines and corresponding corrected profiles in dashed lines.

2.9.4 2D Beam

We consider now a 2D beam of length $L = 16$ m, width $c = 1$ m, Young's modulus $E = 1 \cdot 10^7$ and Poisson's ratio $\nu = 0.3$, initially at rest, assuming no body force and following a 2D linear isotropic elasticity theory with boundary conditions given for all t as follows:

$$\begin{aligned}
u_1(0, y) = u_2(0, y) &= 0, & y \in [0, c], \\
h(x, y) &= (0, 0), & x \in (0, L), y \in [0, c], \\
h_2(L, y) &= 0, & y \in [0, c], \\
h_1(L, y) &= \frac{y - .25}{.25}U, & y \in [.25, .45], \\
h_1(L, y) &= \left(1 - \frac{y - .45}{.25}\right)U, & y \in [.45, .75],
\end{aligned} \tag{2.34}$$

where $U = U(t)$ is the compressive force applied at the free end of the beam and h is the traction. The boundary conditions specify that the left end of the beam is fixed and a force is applied to the right end, distributed so that the maximum force is applied at $y = 0.45$ m, decaying linearly until $y = 0.25$ m and $y = 0.75$ m.

We want to synthesize an input force such that the beam buckles. For this specific setup, we only need to specify the vertical displacement profile in part of the bottom boundary. We formulate the following S-STL specification:

$$\begin{aligned}
\phi &= \mathbf{G}_{[3.45, 4.05]}(A \wedge B), \\
A &= \forall x \in \{x \in \Omega \mid 8 \leq x_1 \leq 14, x_2 = 0\} (u_2(x) > \mu_A(x)), \\
B &= \forall x \in \{x \in \Omega \mid 8 \leq x_1 \leq 14, x_2 = 0\} (u_2(x) < \mu_B(x)),
\end{aligned} \tag{2.35}$$

where μ_A and μ_B are the quadratic functions depicted in Fig. 2·7a (labeled A and B). The specification states that the vertical displacement at part of the bottom boundary must be within the two given profiles at all times between 3.45 s and 4.05 s. Note that a specification defining the shape of a material such as this one can be automatically constructed from a target shape plus a maximum allowed deviation.

We used a regular 8x4 9-node quadratic element mesh and integrated the second order system using the trapezoidal rule with $\Delta t = 75$ ms. The synthesized $U(t)$ is shown in Fig. 2·7d and corresponds to a trajectory with a robustness degree of at least 13. We show snapshots of the evolution of the beam shape in Fig. 2·7. It took

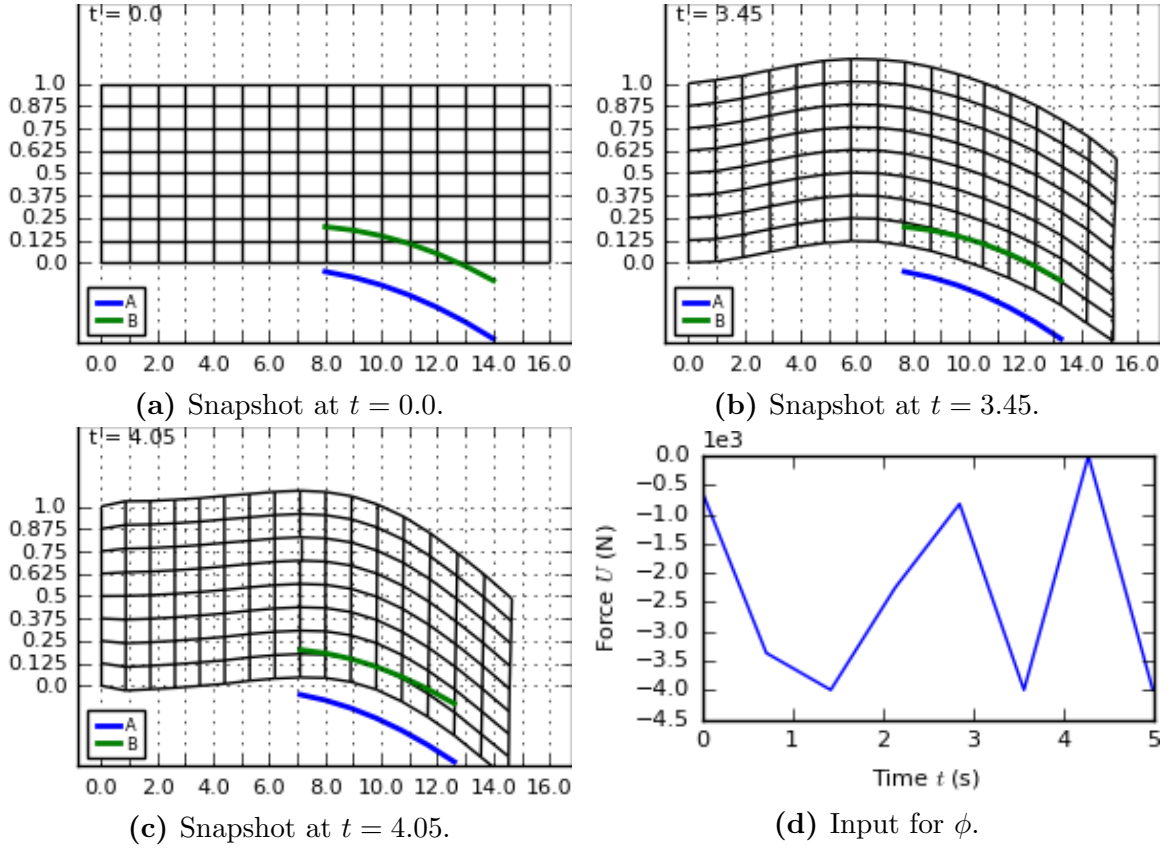


Figure 2.7: Snapshots of the satisfying trajectory for ϕ and synthesized input. The trajectory is shown as the deformation of the domain, represented in black by the mesh used in the FEM. The predicate profiles are shown in colored lines.

5546 seconds to solve the problem.

2.9.5 Nonlinear Elastic Wave Propagation

Here we show a simple synthesis problem similar to the one presented in Sec. 2.9.3 with a nonlinear wave equation of the type described in Sec. 2.7 with the constitutive law (2.27). In order to avoid numerical issues in the MILP, we use the following unitless parameters for the equation: $L = 10$, $\rho = 0.1$, $\epsilon_y = 0.1$, $E_1 = 100$, $E_2 = 50$. The FEM approximation uses a uniform partition with 10 elements and the time interval is $\Delta t = 0.01$. For simplicity, we do not include any correction terms in

the specification. The Neumann control input $U(t)$ is constrained to $[-100, 100]$ with interpolation times $\{0.5i | i = 0, 1, \dots\}$. The target specification states that the displacement must never exceed 0.5 in the interval $[6, 10]$ and at some time between 0.1 and 0.2 the displacement should exceed a linear profile in the interval $[4, 6]$. The corresponding S-STL formula is:

$$\begin{aligned} \phi = & \mathbf{G}_{[0.0,0.2]}(\forall x \in [6, 10] : u(x) < 0.5) \wedge \\ & \mathbf{F}_{[0.1,0.2]}(\forall x \in [4, 6] : u(x) > 0.02u(x) - 0.07). \end{aligned} \quad (2.36)$$

The synthesized inputs are shown in Fig. 2-8, resulting in a trajectory with robustness 0.034. A snapshot of the displacement and strain when it satisfies the second part of the specification is also shown in the figure. It took 8271 seconds to solve the problem.

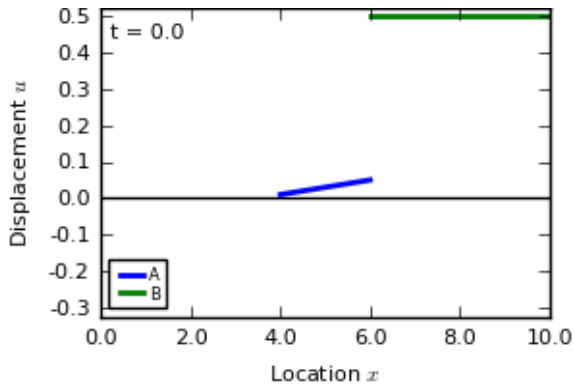
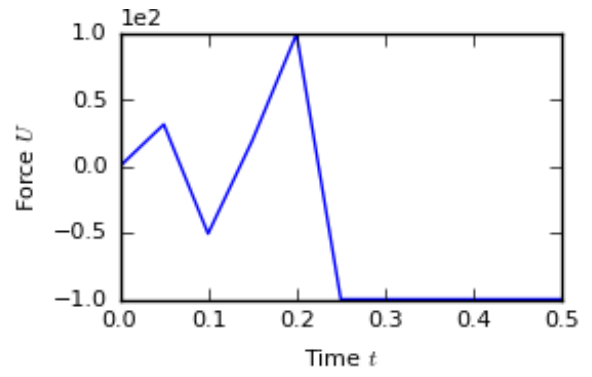
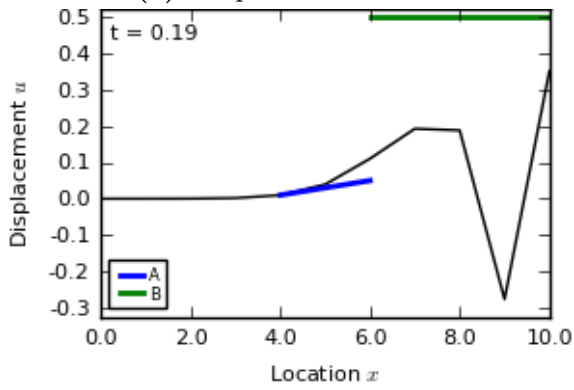
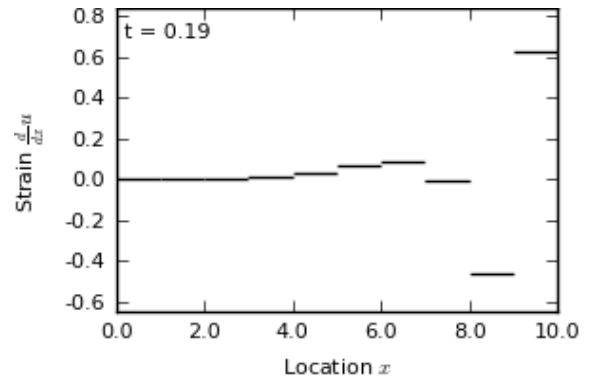
(a) Snapshot at $t = 0.0$.(b) Input for ϕ .(c) Snapshot at $t = 0.19$.(d) Strain at $t = 0.19$.

Figure 2:8: Snapshots of the satisfying trajectory of the nonlinear wave equation for specification ϕ and synthesized input.

Chapter 3

A Decision Tree Approach to Data Classification using Signal Temporal Logic

In this chapter, we present a decision tree based framework for STL inference. The dataset is given as a set of pairs of system traces and labels, where the labels indicate whether the traces exhibit some desired behavior. Our method produces binary decision trees that represent the inferred formula. Each node contains a test associated with the satisfaction of a simple formula, chosen from a set of primitives by optimizing an impurity measure. We extend classical impurity measures with the concept of STL robustness. Our framework is evaluated on anomaly detection problems in maritime security and automotive scenarios.

3.1 Parametric Signal Temporal Logic

Parametric Signal Temporal Logic (PSTL) was introduced in (Asarin et al., 2012) as an extension of STL, where formulae are parameterized. A PSTL formula is similar to an STL formula, however all the time bounds in the time intervals associated with temporal operators and all the constants in the inequality predicates are replaced by free parameters. The two types of parameters are called *time* and *space* parameters, respectively. Specifically, let ψ be a PSTL formula and n_p and n_{TL} be the number of predicates and temporal operators contained in ψ , respectively. The parameter space of ψ is $\Theta = \Pi \times T$, where $\Pi \subseteq \mathbb{R}^{n_p}$ is set of all possible *space* parameters and $T = T_1 \times \dots \times T_{n_{TL}}$ is the set of all *time* parameters, where $T_i = \{(a_i, b_i) \in \mathbb{R}_{\geq 0}^2 \mid a_i \leq b_i\}$

for all $i \in \{1, \dots, n_{TL}\}$. Conversely, if ψ is a PSTL formula, then every parameter assignment $\theta \in \Theta$ induces a corresponding STL formula ϕ_θ , where all the space and time parameters of ψ have been fixed according to θ . This assignment is also referred to as a valuation θ of ψ . For example, given $\psi = \mathbf{G}_{[a,b]}(s_1 \leq c)$ and $\theta = [2.5, 0, 1]$, we obtain the STL formula $\phi_\theta = \mathbf{G}_{[0,1]}(s_1 \leq 2.5)$.

3.2 Problem formulation

We wish to find an STL formula that separates traces produced by a system that exhibit some desired property, such as behaving normally, from other traces of the same system. Formally, let $C = \{C_p, C_n\}$ be the set of classes, with C_p for the positive class and C_n for the negative class. Let s^i be an n -dimensional signal, $s^i : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n$, and let $l^i \in C$ be its label. We consider the following problem:

Problem 3.1 (Two-Class Classification). *Given a set of labeled signals $\{(s^i, l^i)\}_{i=1}^N$, where $l^i = C_p$ if s^i exhibits a desired behavior, and $l^i = C_n$ if s^i does not, find an STL formula ϕ such that the misclassification rate $\text{MCR}(\phi)$ is minimized, where the misclassification rate is defined as:*

$$\text{MCR}(\phi) := \frac{|\{s^i \mid (s^i \models \phi \wedge l^i = C_n) \vee (s^i \not\models \phi \wedge l^i = C_p)\}|}{N}$$

In the above formula, $|\cdot|$ denotes the cardinality of a set, and $(s^i \models \phi \wedge l^i = C_n)$ represents a *false positive*, while $(s^i \not\models \phi \wedge l^i = C_p)$ represents a *false negative*.

3.3 Learning decision trees

In our approach, the key insight to tackle Problem 3.1 is that it is possible to build a map between a fragment of STL and decision trees. Therefore, we can exploit the decision trees learning literature (Ripley, 1996; Quinlan, 2014; Breiman et al., 1984) to build a decision tree that classifies signals and then map the constructed tree to an STL formula.

A decision tree is a tree-structured sequence of questions about the data used to make predictions about the data’s labels. In a tree, we define: the root as the initial node; the depth of a node as the length of the path from the root to that node; the parent of a node as the neighbor whose depth is one less; the children of a node as the neighbors whose depths are one more. A node with no children is called a leaf, all other nodes are called non-terminal nodes. In this paper, we focus on *binary* decision trees, where every non-terminal node splits the data into two children nodes and every leaf node predicts a label.

Unfortunately, the space of all possible decision trees for a given classification problem is very large, and it is known that the problem of learning the optimal decision tree is NP-complete, for various optimality criteria (Hyafil and Rivest, 1976). Therefore, most decision-tree learning algorithms are based on *greedy* approaches, where locally optimal decisions are taken at each node. These greedy growing algorithms can be stated in a simple recursive fashion, starting from the root node, and require three meta-parameters: the first is a list of possible ways to split the data; the second is a criterion to select the best split; and the third is a set of rules for stopping the algorithm.

Several learning algorithms can be created by selecting different meta-parameters. That is, once the meta-parameters have been fixed, a specific learning algorithm is *instantiated*. Since we are not just proposing a single algorithm but a class of algorithms, we refer to this approach as “decision tree learning framework for temporal logic inference”. In the next sections, we explain in detail the parameterized algorithm and the choices we propose for the meta-parameters.

3.3.1 Parameterized learning algorithm

In Alg. 1 we present the parameterized procedure for inferring temporal logic formulae from data. The meta-parameters of Alg. 1 are: (1) a set of PSTL primitives \mathcal{P} ; (2)

an impurity measure J ; and (3) a set of stopping criteria $stop$. The algorithm is recursive and takes as input arguments the formula to reach the current node ϕ^{path} , the set of data that reached that node S , and the current depth level h .

Algorithm 1: Parameterized Decision Tree Construction – $buildTree(\cdot)$

Parameter: \mathcal{P} – set of PSTL primitives
Parameter: J – impurity measure
Parameter: $stop$ – set of stopping criteria
Input: ϕ^{path} – formula associated with current path
Input: $S = \{(s^i, l^i)_{i=1}^N\}$ – set of labeled signals
Input: h – the current depth level
Output: a (sub)-tree

- 1 **if** $stop(\phi^{path}, h, S)$ **then**
- 2 $t \leftarrow leaf(\arg \max_{c \in C} \{p(S, c; \phi^{path})\})$
- 3 **return** t
- 4 $\phi^* = \arg \max_{\psi \in \mathcal{P}, \theta \in \Theta} J(S, partition(S, \phi_\theta \wedge \phi^{path}))$
- 5 $t \leftarrow non_terminal(\phi^*)$
- 6 $S_\top^*, S_\perp^* \leftarrow partition(S, \phi^{path} \wedge \phi^*)$
- 7 $t.left \leftarrow buildTree(\phi^{path} \wedge \phi^*, S_\top^*, h + 1)$
- 8 $t.right \leftarrow buildTree(\phi^{path} \wedge \neg \phi^*, S_\perp^*, h + 1)$
- 9 **return** t

At the beginning, the stopping conditions are checked (line 1). If they are met, the algorithm returns a single leaf node marked with the label $c \in C$. The label c is chosen according to the best classification quality (line 2), using $p(S, c; \phi^{path})$ defined in Def. 3.4. If the stopping conditions are not met (line 4), the algorithm proceeds to find the optimal STL formula among all the valuations of PSTL formulae from the set of primitives \mathcal{P} (details in Sec. 3.3.3). The cost function used in the optimization is the impurity measure J , which assesses the quality of the partition induced by PSTL primitives valuations. See Sec. 3.3.4 for details. At line 5, a new non-terminal node is created and associated with the optimal STL formula ϕ^* . Next, the partition induced by the formula $\phi^{path} \wedge \phi^*$ is computed (line 6). For each outcome of the split, the $buildTree()$ procedure is called recursively to construct the left and right subtrees

(lines 7-8). The corresponding formula to reach a subtree and the corresponding data partition are passed. The depth level is increased by one.

The parameterized family of algorithms uses three procedures: (a) *leaf(c)* creates a leaf node marked with the label $c \in C$, (b) *non_terminal(ϕ)* creates a non-terminal node associated with the valuation of a PSTL primitive from \mathcal{P} , and (c) *partition(S, ϕ)* splits the set of signals S into satisfying and non-satisfying signals with respect to ϕ , i.e., $S_{\top}, S_{\perp} = \text{partition}(S, \phi)$, where $S_{\top} = \{(s^i, l^i) \in S \mid s^i \models \phi\}$ and $S_{\perp} = \{(s^i, l^i) \in S \mid s^i \not\models \phi\}$.

By fixing the meta-parameters $(\mathcal{P}, J, \text{stop})$, a particular algorithm is *instantiated*. For each possible instance, a decision tree is obtained by executing *buildTree($\top, S_{root}, 0$)* on the set of labeled signals S_{root} . Clearly, the returned tree depends on both the input data S_{root} and the particular instance chosen.

3.3.2 Tree to STL formula

A decision tree obtained by an instantiation of Alg. 1 can be used directly for classification or converted to an equivalent STL formula using Alg. 2. The algorithm recursively traverses the subtree t given as input. At each node, the formula is obtained by (1) conjunction of the nodes's formula with its left subtree's formula, (2) conjunction of the negation of the node's formula with its right subtree's formula, (3) disjunction of (1) and (2). During the recursion process, Alg. 2 only keeps track of the paths reaching leaves associated with the positive class C_p . To produce the final formula, the algorithm is executed starting from the root node, i.e., *Tree2STL(root)*. Fig. 3.1 shows a decision tree and its corresponding formula obtained by applying Alg. 2.

Algorithm 2: Tree to formula – $Tree2STL(\cdot)$

Input: t – node of a tree**Output:** STL Formula

```

1 if  $t$  is a leaf and class associated with  $t$  is  $C_p$  then
2   | return  $\top$ 
3 if  $t$  is a leaf and class associated with  $t$  is  $C_n$  then
4   | return  $\perp$ 
5  $\phi_l = (t.\phi \wedge Tree2STL(t.left))$ 
6  $\phi_r = (\neg t.\phi \wedge Tree2STL(t.right))$ 
7 return  $\phi_l \vee \phi_r$ 

```

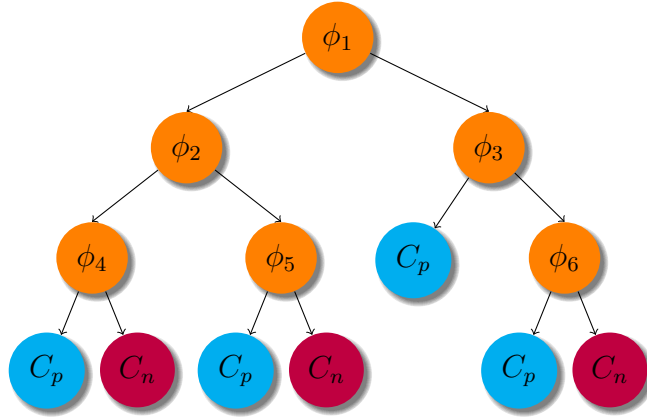


Figure 3-1: The formula associated with the tree is $\phi_{tree} = \left(\phi_1 \wedge \left((\phi_2 \wedge \phi_4) \vee (\neg \phi_2 \wedge \phi_5) \right) \right) \vee \left(\neg \phi_1 \wedge \left(\phi_3 \vee (\neg \phi_3 \wedge \phi_6) \right) \right)$ and can be obtained algorithmically using Alg. 2, where $\phi_i, i \in \{1, \dots, 6\}$ are valuations of primitive formulae from a set of PSTL formulae \mathcal{P} .

3.3.3 PSTL primitives

To partition the data at each node, a finite list of possible splitting rules is usually considered (Ripley, 1996). We propose to use simple PSTL formulae, called *primitives*, to split the data. In particular, we define two types of primitives:

Definition 3.1 (First-Level Primitives). *Let \mathcal{S} be the set of signals with values in $\mathbb{R}^n, n \geq 1$. We define the set of first-level primitives as follows:*

$$\mathcal{P}_1 = \left\{ \mathbf{F}_{[\tau_1, \tau_2]}(x_i \sim \mu) \text{ or } \mathbf{G}_{[\tau_1, \tau_2]}(x_i \sim \mu) \mid i \in \{1, \dots, n\}, \sim \in \{\leq, >\} \right\}$$

The parameters of \mathcal{P}_1 are (μ, τ_1, τ_2) and the space of parameters is $\Theta_1 = \mathbb{R} \times \{(a, b) \mid a < b, a, b \in \mathbb{R}_{\geq 0}\}$.

Definition 3.2 (Second-Level Primitives). *Let \mathcal{S} be the set of signals with values in \mathbb{R}^n , $n \geq 1$. We define the set of second-level primitives as follows:*

$$\mathcal{P}_2 = \left\{ \mathbf{G}_{[\tau_1, \tau_2]} \mathbf{F}_{[0, \tau_3]}(x_i \sim \mu) \text{ or } \mathbf{F}_{[\tau_1, \tau_2]} \mathbf{G}_{[0, \tau_3]}(x_i \sim \mu) \mid i \in \{1, \dots, n\}, \sim \in \{\leq, >\} \right\}$$

The parameters of \mathcal{P}_2 are $(\mu, \tau_1, \tau_2, \tau_3)$ and the space of parameters is $\Theta_2 = \mathbb{R} \times \{(a, b) \mid a < b, a, b \in \mathbb{R}_{\geq 0}\} \times \mathbb{R}_{\geq 0}$.

The meaning of first-level primitives is straightforward. The two primitives $\mathbf{F}_{[\tau_1, \tau_2]}(x_i \sim \mu)$ and $\mathbf{G}_{[\tau_1, \tau_2]}(x_i \sim \mu)$ are used to express that the predicate $x_i \sim \mu$ must be true for at least one time instance or for all time instances in the interval $[\tau_1, \tau_2)$, respectively. Similarly, the second-level primitives can be interpreted in natural language as: (a) $\mathbf{F}_{[\tau_1, \tau_2]} \mathbf{G}_{[0, \tau_3]}(x_i \sim \mu)$ specifies that “the predicate $(x_i \sim \mu)$ of duration τ_3 must be performed and its start time must be in the interval $[\tau_1, \tau_2)$ ”; and (b) $\mathbf{G}_{[\tau_1, \tau_2]} \mathbf{F}_{[0, \tau_3]}(x_i \sim \mu)$ specifies that “at each time instance in the interval $[\tau_1, \tau_2)$, the predicate $(x_i \sim \mu)$ must be true within τ_3 time units”. Both first- and second-level primitives may be thought as specifications for bounded reachability and safety with varying degrees of flexibility.

Given a set of primitives \mathcal{P} , we denote by $\text{STL}_{\mathcal{P}}$ the STL fragment obtained by Boolean closure from \mathcal{P} .

Definition 3.3 (Boolean Closure). *Let \mathcal{P} be a finite set of PSTL formulae. The fragment of STL formulae induced by \mathcal{P} using Boolean closure is defined as:*

$$\phi ::= \top \mid \varphi \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2$$

where φ is a valuation of a PSTL formula from \mathcal{P} .

$\text{STL}_{\mathcal{P}}$ is the fragment of STL that is mapped with decision trees. In other terms,

each decision tree constructed with the set of primitives \mathcal{P} is mapped to an STL formula belonging to the $\text{STL}_{\mathcal{P}}$ fragment.

Remark 3.1. *Note that $\text{STL}_{\mathcal{P}_1} \subset \text{STL}_{\mathcal{P}_2}$, because*

$\mathbf{F}_{[\tau_1, \tau_2]} l \equiv \mathbf{F}_{[\tau_1, \tau_2]} \mathbf{G}_{[0, 0^+]} l$ and similarly $\mathbf{G}_{[\tau_1, \tau_2]} l \equiv$

$\mathbf{G}_{[\tau_1, \tau_2]} \mathbf{F}_{[0, 0^+]} l$, where $l \equiv (x_i \sim \mu)$ is a linear inequality predicate and 0^+ represents the upper limit towards 0.

Remark 3.2. *It is important to stress that the proposed PSTL primitives are not the only possible ones. A user may define other primitives, either generic ones, like the first- and second- level primitives, or specific ones, guided by the particular nature of the learning problem at hand.*

3.3.4 Impurity measures

In the previous section, we defined a list of possible ways to split the data using a set of primitives \mathcal{P} . Now, it is necessary to define a criterion to select which primitive best splits the data at each node. Intuitively, a good split leads to children that are *homogeneous*, that is, they contain mostly objects belonging to the same class. This concept has been formalized in literature with *impurity measures*, and the goal of the optimization algorithm is to obtain children purer than their parents. In this section, we first state the canonical impurity measures and then we propose three modified measures, which are more suited to handle signals, using the robustness degree.

Definition 3.4 (Impurity Measures). *Let S be a finite set of signals, ϕ an STL formula and*

$S_{\top}, S_{\perp} = \text{partition}(S, \phi)$. *The following partition weights are introduced to describe how the signals s^i are distributed according to their labels l^i and the formula ϕ :*

$$p_{\top} = \frac{|S_{\top}|}{|S|}, \quad p_{\perp} = \frac{|S_{\perp}|}{|S|}, \quad p(S, c; \phi) = \frac{|\{(s^i, l^i) \mid l^i = c\}|}{|S|} \quad (3.1)$$

Particularity, p_{\top} and p_{\perp} represent the fraction of signals from S present in S_{\top} and S_{\perp} , respectively, and $p(S, c; \phi)$ represents the fraction of signals in S that belong to class $c \in C$.

The (canonical) impurity measures are defined as (Breiman et al., 1984; Quinlan, 2014):

- Information gain (IG)

$$\begin{aligned} IG(S, \{S_{\top}, S_{\perp}\}) &= H(S) - \sum_{\otimes \in \{\top, \perp\}} p_{\otimes} \cdot H(S_{\otimes}) \\ H(S) &= - \sum_{c \in C} p(S, c; \phi) \log p(S, c; \phi) \end{aligned} \quad (3.2)$$

- Gini gain (GG)

$$\begin{aligned} GG(S, \{S_{\top}, S_{\perp}\}) &= Gini(S) - \sum_{\otimes \in \{\top, \perp\}} p_{\otimes} \cdot Gini(S_{\otimes}) \\ Gini(S) &= \sum_{c \in C} p(S, c; \phi) (1 - p(S, c; \phi)) \end{aligned} \quad (3.3)$$

- Misclassification gain (MG)

$$\begin{aligned} MG(S, \{S_{\top}, S_{\perp}\}) &= MR(S) - \sum_{\otimes \in \{\top, \perp\}} p_{\otimes} \cdot MR(S_{\otimes}) \\ MR(S) &= \min(p(S, C_p; \phi), p(S, C_n; \phi)) \end{aligned} \quad (3.4)$$

We extend the impurity measures to account for the robustness degrees of the signals to be classified. These extensions are based on the intuition that, according to Prop. 2.1, the robustness degree can be used in the context of learning as a measure of the classification quality of a signal with respect to an STL formula.

Definition 3.5 (Extended Impurity Measures).

Consider the same setup as in Def. 3.4, and the same impurity measures, we redefine the partition weights as follows:

$$\begin{aligned} p_{\top} &= \frac{\sum_{s^i \in S_{\top}} r(s^i, \phi)}{\sum_{s^i \in S} |r(s^i, \phi)|} & p_{\perp} &= - \frac{\sum_{s^i \in S_{\perp}} r(s^i, \phi)}{\sum_{s^i \in S} |r(s^i, \phi)|} \\ p(S, c; \phi) &= \frac{\sum_{s^i \in S_c} |r(s^i, \phi)|}{\sum_{s^i \in S} |r(s^i, \phi)|} \end{aligned} \quad (3.5)$$

where $S_c = \{s^i \in S \mid l^i = c\}$.

We will distinguish between the usual impurity measures and the extended ones by using the subscript r (e.g., IG_r) for the extended impurity measures. The following proposition ensures that the extended impurity measures are well defined.

Proposition 3.1. *The intra-partition weights are bounded within 0 and 1 and sum to 1, i.e., $0 \leq p_{\top}, p_{\perp} \leq 1$ and $p_{\top} + p_{\perp} = 1$, in both definitions Def. 3.4 and Def. 3.5. The same invariant property is true for the inter-partition weights, i.e., $0 \leq p(S, C_n; \phi), p(S, C_p; \phi) \leq 1$ and $\sum_{c \in C} p(S, c; \phi) = 1$.*

Proof. For Def. 3.4, the result is immediate as S_{\top}, S_{\perp} define a partition of S . In the case of the extended impurity measures in Def. 3.5, note that the robustness of signals in S_{\top} is positive and negative for those in S_{\perp} by definition of those sets. \square

Remark 3.3. *The advantages of using the extended versions of the impurity measures over the canonical ones are most pertinent in the context of optimizing these over PSTL formulae. The robustness-based impurity functions are better behaved cost functions, because these are less flat over the space parameter than their frequency-based counterparts, i.e., the canonical measures are piecewise constant functions. Also, we argue that the use of robustness makes the computed classifiers better at generalizing, i.e., performance on unseen (test) data. The intuition is that the separation boundaries tend to be as far as possible from signals of the two classes in the sense of robustness. In this sense, the canonical measures are unable to distinguish between formulae which are barely satisfied by some signals from more robust ones. As a future work, an empirical comparison of the robustness-based measures against the canonical ones would strengthen this discussion.*

Local optimization

The cost function used in the local node optimization (line 4 of Alg. 1) is one of the impurity measures defined in the previous section. The optimization is performed over the chosen set of PSTL primitives \mathcal{P} and their valuations Θ . Therefore, the optimization problem is decomposed into $|\mathcal{P}|$ optimization problems over a fixed and small number of real-valued parameters. Consider signals of dimension n . In the case of \mathcal{P}_1 , we have $4n$ optimization problems with 3 parameters each. On the other hand, for \mathcal{P}_2 we have $4n$ optimization problems with 4 parameters each.

The local optimization approach presents several advantages. In particular, the computation of the robustness values in the definition of the extended impurity measures (Def. 3.5) can be performed incrementally with respect to the tree data structure according to the following proposition.

Proposition 3.2. (INCREMENTAL COMPUTATION OF ROBUSTNESS) *At each step of the recursion of Alg. 1, the robustness of a signal s^i reaching the current node n_c can be computed as follows*

$$r(s^i, \phi^{tree}) = r(s^i, \phi^{path} \wedge \phi) = \min\{r(s^i, \phi^{path}), r(s^i, \phi)\} \quad (3.6)$$

where ϕ^{tree} corresponds to the currently computed tree, ϕ^{path} corresponds to the branch of the tree from the root to the parent of n_c , and ϕ is a candidate valuation of a PSTL primitive for n_c .

Proof. By construction (see recursion lines), $\phi^{tree} = \phi^{path} \wedge \phi$. By definition of robustness, $r(s^i, \phi^{path} \wedge \phi) = \min\{r(s^i, \phi^{path}), r(s^i, \phi)\}$. \square

The first equality in Eq. (3.6) follows from the construction of the tree, because the robustness of a signal s^i reaching n_c is negative for any other branch of the tree not ending in n_c . The incremental computation can be achieved by taking advantage of the recursion in the second equality in Eq. (3.6).

Another very important advantage of the proposed approach is that at each iteration of Alg. 1, the data is partitioned between the children of the currently processed node. Thus, the local optimization problems become easier as the depth of the nodes increases.

The local optimization problems may be solved using any global non-linear optimization algorithm, such as Simulated Annealing or Differential Evolution. However, in order to use these numerical optimization algorithms, we need to define finite bounds for the parameters of the primitive formulae. These bounds may easily be inferred from data, but may also be application specific, if expert knowledge is available.

3.3.5 Stop conditions

Several stopping criteria can be set for Alg. 1. The most common strategy is to just split until the current node contains only signals from a single class or no signals. This strategy is very permissive, that is, it allows the algorithm to run for many iterations. However, it represents the sufficient conditions that guarantee the termination of the algorithm. Other more restrictive conditions are possible. For instance, stop if the vast majority of the signals belong to the same class, either positive or negative, e.g., stop if 99% of signals belong to the same class. Another common strategy is to stop if the algorithm has reached a certain, fixed, depth. These conditions usually provide a faster termination of the algorithm. In general, a set of stopping criteria can be assembled by picking several stopping conditions, as long as the sufficient conditions for the termination of the algorithm are included.

3.3.6 Complexity

In this section, we provide a worst-case and average-case complexity analysis of Alg. 1 in terms of the complexity of the local optimization procedure (Alg. 1, line 4). This complexity analysis assumes that just the sufficient stopping conditions are set. Let $C(N)$ and $g(N)$ be the complexity of Alg. 1 and of the local optimization algorithm, respectively, where N is the number of signals to be processed by the algorithms. Trivially, we have $g(N) = \Omega(N)$, where $\Omega(\cdot)$ is the asymptotic notation for lower bound (Cormen, 2009), because the algorithm must at least check the labels of all signals. The worst-case complexity of Alg. 1 is attained when at each node the optimal partition has size $(1, N - 1)$. In this case, the complexity satisfies the recurrence $C(N) = C(N-1) + C(1) + g(N)$, which implies $C(N) = \Theta(N + \sum_{k=2}^N g(k))$, where $\Theta(\cdot)$ is the two-sided asymptotic notation for complexity bound (Cormen, 2009). However, the worst case scenario is not likely to occur in large datasets. Therefore, we consider

the average case where at least a fraction $\gamma \in (0, 1)$ of the signals are in one set of the partition. The recurrence relation becomes $C(N) = C(\gamma N) + C((1 - \gamma)N) + g(N)$, which implies the following complexity bound

$$C(N) = \Theta \left(N \cdot \left(1 + \int_1^x \frac{g(u)}{u^2} \mathrm{d}u \right) \right)$$

obtained using the Akra-Bazzi method (Cormen, 2009). Finally, note that the hidden constants in the complexity bounds above depend on the cardinality of the set of primitives considered and the size of their parameterization.

3.4 Case Studies

In this section, we present two case studies that illustrate the usefulness and the computational advantages of the algorithms. The first is an anomalous trajectory detection problem in a maritime environment. The second is a fault detection problem in an automotive powertrain system. The automotive application is particularly appealing because the systems involved are getting more and more sophisticated. In a modern vehicle, several highly complex dynamical systems are interconnected and the methods present in literature may fail to cope with this complexity.

3.4.1 Maritime surveillance

This synthetic dataset emulates a maritime surveillance problem, where the goal is to detect suspicious vessels approaching the harbor from sea by looking at their trajectories. It was developed in (Kong et al., 2017), based on the scenarios described in (Kowalska and Peel, 2012), for evaluating their inference algorithms.

The trajectories are represented with planar coordinates $x(t)$ and $y(t)$ and were generated using a Dubins' vehicle model with additive Gaussian noise. Three types of scenarios, one normal and two anomalous, were considered. In the normal scenario, a

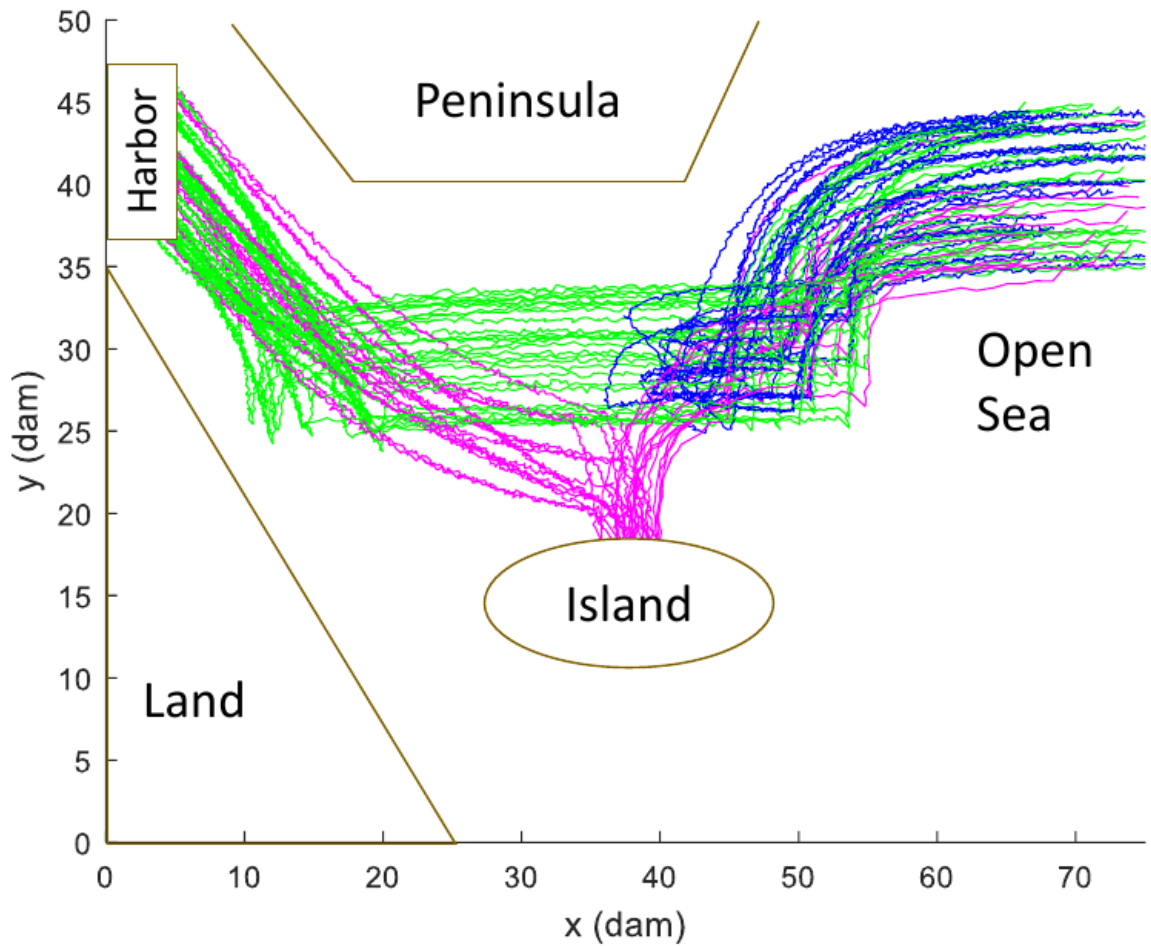


Figure 3-2: Naval surveillance dataset (Kong et al., 2017). The vessels behaving normally are shown in green. The magenta and blue trajectories represent two types of anomalous paths.

vessel approaching from sea heads directly towards the harbor. In the first anomalous scenario, a ship veers to the island and heads to the harbor next. This scenario is compatible with human trafficking. In the second anomalous scenario, a boat tries to approach other vessels in the passage between the peninsula and the island and then veers back to the open sea. This scenario is compatible with terrorist activity. Some sample traces are shown in Fig. 3-2. The dataset is composed of 2000 total traces, with 61 sample points per trace. There are 1000 normal traces and 1000 anomalous.

3.4.2 Fuel control system

We investigate a fuel control system for a gasoline engine. A model for this system is provided as built-in example in Simulink and we modified it for our purposes. This model was initially used for Bayesian statistical model checking (Zuliani et al., 2013) and has been recently proposed as benchmark for the hybrid systems community (Hoxha et al., 2014). We selected this model because it includes all the complexities of real world industrial models, but is still quick to simulate, i.e., it is easy to obtain a large number of traces.

The key quantity in the model is the *air-to-fuel ratio*, that is, the ratio between the mass of air and the mass of fuel in the combustion process. The goal of the control system is to keep it close to the “ideal” stoichiometric value for the combustion process. For this system, the target air-fuel ratio is 14.6, as it provides a good compromise between power, fuel economy, and emissions. The system has one main output, the air-to-fuel ratio, one control variable, the fuel rate, and two inputs, the engine speed and the throttle command. The system estimates the correct fuel rate to achieve the target stoichiometric ratio by taking into account four sensor readings. Two are related directly to the inputs, the engine speed and the throttle angle. The remaining two sensors provide crucial feedback information: the EGO sensor reports the amount of residual oxygen present in the exhaust gas, and the MAP sensor reports the (intake) manifold absolute pressure. The EGO value is related to the air-to-fuel ratio, whereas the MAP value is related to the air mass rate. The Simulink diagram is made of several subsystems with different kinds of blocks, both continuous and discrete, among which there are look-up tables and a hybrid automaton. Due to these characteristics, this model can exhibit a rich and diverse number of output traces, thus making it an interesting candidate for our investigation.

The base model, that is, the one included in Simulink, includes a very basic fault

detection scheme and fault injection mechanism. The fault detection scheme is a simple threshold crossing test (within a Stateflow chart), and is only able to detect single off range values. For avoiding the overlap of two anomaly detection schemes, the built-in one has been removed. In the base model, the faults are injected by simply reporting an incorrect and fixed value for a sensor's reading. Moreover, these faults are always present from the beginning of the simulation. We replaced this simple fault injection mechanism with a more sophisticated unit. The new subsystem is capable of inducing faults in both the EGO and MAP sensors with a *random* arrival time and with a *random* value. Specifically, the faults can manifest at anytime during the execution (uniformly at random) and the readings of the sensors affected are offset by a value that *varies* at every execution. Finally, independent Gaussian noise signals, with zero mean and variance $\sigma^2 = 0.01$, have been added at the output of the sensors.

For the fuel control system, 1200 total simulations were performed. In all cases, the throttle command provides a periodic triangular input, and the engine speed is kept constant at 300 rad/sec (2865 RPM). The simulation time is 60 seconds. In details, we obtained: 600 traces where the system was working normally; 200 traces with a fault in the EGO sensor; 200 traces with a fault in the MAP sensor; 200 traces with faults in both sensors. For every trace, we collected 200 samples of the EGO and MAP sensors' readings. Some sample traces are shown in Fig. 3-3. The average simulation time to obtain a single trace was roughly 1 second.

3.5 Implementation and results

We implemented and tested two different instances of Alg.1, I_1 and I_2 , defined by the choice of meta-parameters given in Table 3.1. In the case of I_1 , the implementation was done in MATLAB using standard libraries, employing the simulated annealing optimization method, and run on a 3.5 GHz processor with 16 GB RAM. As for

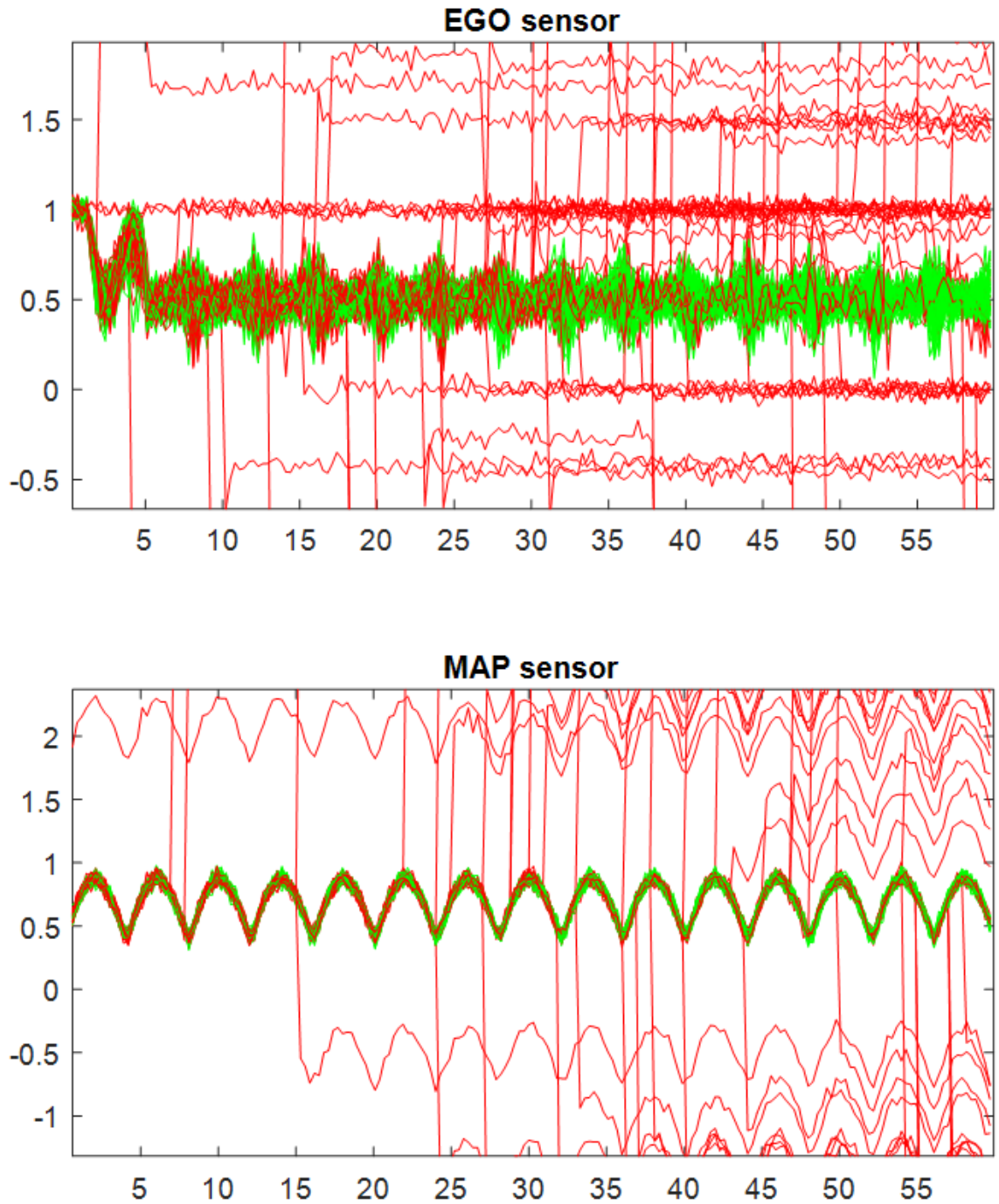


Figure 3-3: Fuel Control Dataset. Normal traces are shown in green, anomalous traces are shown in red.

Instance	Primitives	Impurity	Stopping
I_1	\mathcal{P}_1	MG_r	Majority class rate > 0.975 , Depth > 4
I_2	\mathcal{P}_2	IG_r	Depth > 3

Table 3.1: Algorithm meta-parameters. See Sec. 3.3 for details.

I_2 , we used the SciPy library for Python, solving the optimization problem with its implementation of the differential evolution algorithm, and we tested it on similar hardware.

3.5.1 Maritime surveillance

We tested the I_2 instance using a non stratified 10-fold cross-validation with a random permutation of the data set, obtaining a mean misclassification rate of 0.007 with a standard deviation of 0.008 and a run time of about 4 hours per split. A sample formula learned in one of the cross-validation splits is:

$$\begin{aligned}
\phi^{I_2} &= (\phi_1^{I_2} \wedge (\neg\phi_2^{I_2} \vee (\phi_2^{I_2} \wedge \neg\phi_3^{I_2}))) \vee (\neg\phi_1^{I_2} \wedge (\phi_4^{I_2} \wedge \phi_5^{I_2})) \\
\phi_1^{I_2} &= \mathbf{G}_{[199.70,297.27]} \mathbf{F}_{[0.00,0.05]}(x \leq 23.60) \\
\phi_2^{I_2} &= \mathbf{G}_{[4.47,16.64]} \mathbf{F}_{[0.00,198.73]}(y \leq 24.20) \\
\phi_3^{I_2} &= \mathbf{G}_{[34.40,52.89]} \mathbf{F}_{[0.00,61.74]}(y \leq 19.62) \\
\phi_4^{I_2} &= \mathbf{G}_{[30.96,37.88]} \mathbf{F}_{[0.00,250.37]}(x \leq 36.60) \\
\phi_5^{I_2} &= \mathbf{G}_{[62.76,253.23]} \mathbf{F}_{[0.00,41.07]}(y \leq 29.90)
\end{aligned} \tag{3.7}$$

We can see in Fig. 3.4 how the thresholds for ϕ_1 and ϕ_2 capture the key features of the data set. Notice also the insight we can gain from their plain English translation: “Normal vessels’ x coordinate is below 23.6 during the last 100 seconds, i.e., they approach and remain at the port”, and “normal vessels’ y coordinate never go below 24.2, i.e., they don’t approach the island”. It is worth mentioning the second term of the outer disjunction in ϕ^{I_2} , as it highlights a feature of the data set difficult to spot on the figures: some normal vessels don’t reach the port (inspecting the data set,

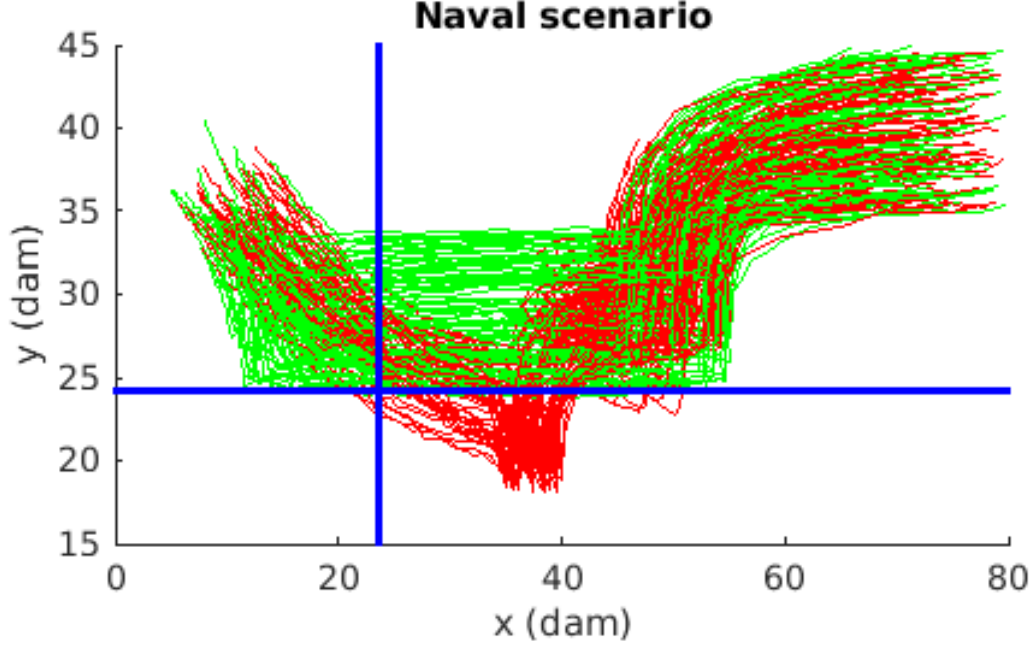


Figure 3-4: Sample of the naval surveillance dataset. Normal trajectories are green and anomalous trajectories are red. We show in blue the boundaries of $\phi_1^{I_2}$ and $\phi_2^{I_2}$ of Eq. (3.7).

some normal traces stop right after crossing the passage). As usual when employing decision trees, deeper formulae focus on finer details of the data set.

In the case of I_1 , we tested it using a 5-fold cross-validation, obtaining a mean misclassification rate of 0.01 and a standard deviation of 0.0064. The run time is about 16 minutes per split. A sample formula learned in one of the splits is:

$$\begin{aligned}
 \phi^{I_1} &= (\phi_1^{I_1} \wedge (\phi_2^{I_1} \wedge \phi_3^{I_1})) \vee (\neg \phi_1^{I_1} \wedge (\phi_4^{I_1} \wedge \phi_5^{I_1})) \\
 \phi_1^{I_1} &= \mathbf{G}_{[94.6, 300)}(y \leq 35.3) \\
 \phi_2^{I_1} &= \mathbf{G}_{[0, 300)}(y > 23) \\
 \phi_3^{I_1} &= \mathbf{G}_{[298, 300)}(x \leq 25.9) \\
 \phi_4^{I_1} &= \mathbf{G}_{[182, 300)}(x \leq 19.6) \\
 \phi_5^{I_1} &= \mathbf{G}_{[0, 51.6)}(x > 42.6)
 \end{aligned} \tag{3.8}$$

Note the similarity between the subformulae $\phi_2^{I_2}$ and $\phi_2^{I_1}$, or between $\phi_1^{I_2}$ and $\phi_3^{I_1}$ in Eq. (3.7) and Eq. (3.8), respectively.

This dataset was also used in (Kong et al., 2017). Unfortunately, it is not possible to make a formal comparison between the formulae learned by our approach and the ones in (Kong et al., 2017). This is due to the fact that iPSTL, defined in (Kong et al., 2017), and $\text{STL}_{\mathcal{P}_1}$ (or $\text{STL}_{\mathcal{P}_2}$) do not represent the same STL fragment. However, it is always possible to make a comparison in terms of sheer classification performance. In the comparison, it is clear that we improve the misclassification rate by a factor of 20 while spending a similar amount of execution time.

3.5.2 Fuel control

In this scenario, we tested both instances using the EGO and MAP sensors' readings (variables x_1 and x_2). We performed a similar cross-validation for I_2 , resulting in a mean misclassification rate of 0.054 with a standard deviation of 0.025 and a run time of about 15 hours per split. A sample formula, obtained from one of the cross-validation splits, is:

$$\begin{aligned}
 \phi^{I_2} &= \neg\phi_1^{I_2} \wedge \phi_2^{I_2} \wedge \phi_3^{I_2} \\
 \phi_1^{I_2} &= \mathbf{F}_{[1.85,58.70)} \mathbf{G}_{[0.00,0.57)}(x_1 \leq 0.13) \\
 \phi_2^{I_2} &= \mathbf{G}_{[11.35,59.55)} \mathbf{F}_{[0.00,0.03)}(x_1 \leq 0.99) \\
 \phi_3^{I_2} &= \mathbf{G}_{[1.65,58.89)} \mathbf{F}_{[0.00,0.44)}(x_2 \leq 0.90)
 \end{aligned} \tag{3.9}$$

Notice in this case how the resulting subformulae are equivalent to first-level primitives, suggesting that \mathcal{P}_2 is an overly complicated set of primitives.

Regarding I_1 , using a 5-fold cross-validation, we obtained a mean misclassification rate of 0.075 and a standard deviation of 0.0256. The run time is about 18 minutes

per split. A sample formula learned in one of the splits is:

$$\begin{aligned}
\phi^{I_1} &= \phi_1^{I_1} \wedge (\phi_2^{I_1} \wedge (\phi_3^{I_1} \wedge \phi_4^{I_1})) \\
\phi_1^{I_1} &= \mathbf{G}_{[0,59.7)}(x_2 > -0.563) \\
\phi_2^{I_1} &= \mathbf{G}_{[0,59.7)}(x_2 \leq 1.91) \\
\phi_3^{I_1} &= \mathbf{G}_{[0,59.7)}(x_1 > -0.819) \\
\phi_4^{I_1} &= \mathbf{G}_{[23.7,59.7)}(x_1 \leq 1.78)
\end{aligned} \tag{3.10}$$

In both case studies, the execution time of I_2 is higher than I_1 . This occurs because the instance I_2 involves a more complicated optimization problem. Specifically, I_2 uses primitives from \mathcal{P}_2 with 4 free parameters, whereas I_1 uses primitives with only 3 free parameters.

Chapter 4

Assumption Mining

In this chapter, we show how an Assume-Guarantee Contract (AGC) approach can be used to address the scalability issues of our solution to the boundary control synthesis problem described in the first part of the dissertation. Then, we formulate the related subproblem of assumption mining and propose a sampled based algorithm to solve it. Our method produces an approximation to the set of assumptions by iteratively sampling from within and without the set of assumptions. From these samples, an STL formula representing the current approximation is inferred. A tolerance controlling the quality of the approximation can be set and we provide an explicit bound on this quality in terms of the tolerance. A simple case study is presented to illustrate the algorithm.

4.1 Assume-Guarantee Contracts and Assumption Mining

In this section we formalize the assumption mining problem. As motivation, we first formalize the distributed control synthesis problem and how it can be solved through the use of assume-guarantee contracts. Assumption mining is the first step towards the synthesis of assume-guarantee contracts for a distributed system. Note that we do not solve the distributed control synthesis problem in this work.

Let $\Sigma(u)$ be the discrete-time system described by the following relation:

$$\Sigma(x_0, u) : x_{k+1} = Ax_k + Bu_k + F, \quad (4.1)$$

where $x_i \in \Omega \subset \mathbb{R}^n, u_i \in U \subset \mathbb{R}^m, u = u_0 u_1 \dots$ and x_0 is given. Let the system be partitioned in l subsystems, $\{\Sigma^i\}_{i=1}^l$, where each subsystem has the following form:

$$\Sigma^i(u) : x_{k+1}^i = A^i x_k^i + B^i u_k^i + \sum_{j \neq i} A^{j \rightarrow i} x_k^{j \rightarrow i} + F^i, \quad (4.2)$$

where the state vector x and control vector u have been partitioned (i.e., there are no shared states or controls), and the system matrices have been rewritten accordingly. Note that $x^{j \rightarrow i}$ refers to the set of state variables corresponding to subsystem j that have direct influence in subsystem i .

Consider an STL formula $\phi = \bigwedge_{i=1}^l \phi^i$, where ϕ^i is an STL formula over x^i . We define the distributed control synthesis problem as the following:

Problem 4.1 (Distributed control synthesis). *Find a control policy $u = u_0 u_1 \dots$, with u_t^i dependent on the state and control history of subsystem i , such that $\Sigma(u) \models \phi$.*

An Assume-Guarantee Contract (AGC) is an STL formula $\psi^{i \rightarrow j}$ over $x^{i \rightarrow j}$. Our objective is to find a set of AGCs such that there exists local control policies u^i satisfying the following property:

$$\forall i = 1, \dots, l : (\forall j \neq i : \Sigma^j(u) \models \psi^{j \rightarrow i}) \implies \Sigma^i(u) \models \phi^i \wedge \left(\bigwedge_{j \neq i} \psi^{i \rightarrow j} \right). \quad (4.3)$$

In other words, if a subsystem has all its assumptions guaranteed, then it can satisfy the local specification and its guarantees. We call a set of contracts satisfying this property well posed.

The relevance of Problem 4.1 to our work is the following: suppose we have a boundary control synthesis problem for a PDE system. If the discretized system and S-STL specification (as defined in Chapter 2) can be decomposed in discrete time subsystems and local STL specifications as described above, then we can solve the boundary control synthesis problem by solving a distributed control synthesis problem. The use of AGCs has been proven useful in the solution of Problem 4.1, see

for example (Sadraddini et al., 2017).

We move now to assumption mining, a first step in the synthesis of well posed AGCs. Let $\Sigma(x_0, z)$ be the discrete-time system described by the following relation:

$$\Sigma(x_0, z) : x_{k+1} = Ax_k + Dz_k + F, \quad (4.4)$$

where $x_i \in \Omega \subset \mathbb{R}^n, z_i \in Z \subset \mathbb{R}^m, z = z_0 z_1 \dots$. Let ϕ be an STL formula over x . We define the assumption mining problem as the following:

Problem 4.2 (Assumption mining). *Find an STL formula ϕ_a over x_0 and z such that if $(x_0, z) \models \phi_a$ then $\Sigma(x_0, z) \models \phi$. In addition, for any other STL formula ϕ'_a such that $\phi'_a \implies \phi_a$ and they are not logically equivalent, if $(x_0, z) \models \phi'_a$ and $(x_0, z) \not\models \phi_a$ then $\Sigma(x_0, z) \not\models \phi$.*

In the following, we require the following assumption:

Assumption 4.1. *The subsets Ω and Z are bounded.*

Our solution to the assumption mining problem is an adaptation of the sampled based algorithm found in (Kim et al., 2016) such that it does not assume any kind of monotonicity of the system. This assumption does not generally hold in discretized PDE systems, which prevents us from using the cited algorithm as a basis for solving the distributed control synthesis problem for PDEs. When the monotonicity assumption is not met, the set of assumptions is no longer possible to describe as an STL formula of special form (so called directed specifications), which can be easily constructed from a set of valid and invalid assumptions. Instead, we must use a general inference algorithm that can provide us with an STL formula that describes the sets of assumption samples.

4.2 Online Decision Tree STL Inference

In order to produce an STL formula from a set of assumption samples, an STL inference algorithm such as the one described in Chapter 3 must be used. Since

the assumption mining algorithm will incrementally collect samples and obtain STL formulas representing them, we present here a simple online version of the STL inference algorithm. For a thorough discussion on online STL inference refer to (Bombara, 2020).

We will instantiate the STL inference algorithm with the following meta-parameters: let \mathcal{P} be the set of first-level primitives described in Def. 3.1, J the extended information gain impurity measure IG_r , and *stop* the perfect classification stopping criteria (i.e., the recursive algorithm only stops once a tree that perfectly classifies the training set is constructed). In the online version shown as pseudocode in Algorithm 3, we start by constructing a decision tree with the initial training set. As a new sample is received, we find the leaf corresponding to the sample and run the recursive step on the leaf. If it is impure, it will be split until perfect classification is achieved (the stopping condition). Otherwise, the tree remains unchanged. After a set amount of new traces has been incorporated to the training set, the current tree is discarded and a new one is built.

4.3 Sampling Based Assumption Mining

In this section we present an algorithm that computes an ϵ -approximation to the solution of Problem 4.2, ϕ_a . Algorithm 4 obtains an approximation $\tilde{\phi}_a$ to ϕ_a such that $\tilde{\phi}_a \implies \phi_a$, i.e., it represents an underapproximation to the set of admissible traces of ϕ_a . Once an underapproximation is obtained, the algorithm improves the approximation by growing the set of admissible traces of $\tilde{\phi}_a$. If at some point the formula admits traces that should not be in ϕ_a , the approximation is shrunk until it becomes an underapproximation again.

The approximations are computed using a mix of sampling-based techniques, machine learning and MILP solving. At every iteration of the algorithm, we attempt

Algorithm 3: Online Decision Tree Construction – $fitNewSample(\cdot)$

Parameter: k – samples to process online before rebuilding tree
Input: t_{cur} – current tree
Input: (s, l) – new labeled signal (sample)
Output: the new tree

```

1 if  $t_{cur}.newsamples > k$  then
2    $t_{cur} \leftarrow buildTree(\top, t_{cur}.signals \cup \{(s, l)\})$ 
3    $t_{cur}.newsamples \leftarrow 0$ 
4   return  $t_{cur}$ 
5  $t \leftarrow FindLeaf(t_{cur}, s)$ 
6 if  $Classify(t_{cur}, s) \neq l$  then
7    $t' \leftarrow buildTree(Path(t), t.signals \cup \{(s, l)\}, Depth(t) + 1)$ 
8    $SubstituteNode(t_{cur}, t, t')$ 
9 else
10   $t.signals \leftarrow t.signals \cup \{(s, l)\}$ 
11  $t_{cur}.newsamples \leftarrow t_{cur}.newsamples + 1$ 
12 return  $t_{cur}$ 

```

to find a trace satisfying $\tilde{\phi}_a$ such that $\Sigma(x_0, z) \not\models \phi$. This is done by solving the optimization problem posed in Line 4, which can be encoded as an MILP and solved using off-the-shelf tools. If one is found, we keep that sample as an unsatisfying sample (Line 6). Otherwise, we attempt to find a trace not satisfying $\tilde{\phi}_a$ such that $\Sigma(x_0, z) \models \phi$ (Line 8) and keep it as a satisfying sample (Line 10). If a sample was added, we recompute $\tilde{\phi}_a$ using a temporal inference tool (Line 3). The resulting formula is satisfied by all satisfying samples and not satisfied by all unsatisfying samples.

In order to improve the performance of the algorithm, we “bloat” $\tilde{\phi}_a$ and its negation when we solve the optimization problems. We do this by constraining the samples to be satisfying with robustness larger than a positive real number ϵ . This also ensures the algorithm doesn’t get stuck changing the size of the set of satisfying traces by vanishing amounts. If no new sample is found in an iteration of the algorithm, we decrease ϵ using a learning parameter α (Line 12) until we reach a minimum tolerance.

The quality of the approximation is established in the following theorem:

Theorem 4.1. *Algorithm 4 finishes in finite time and the returned $\tilde{\phi}_a$ satisfies the following properties:*

1. *If $\rho(\tilde{\phi}_a, (x_0, z)) > \epsilon_0$ then $\Sigma(x_0, z) \models \phi$.*
2. *If $\rho(\tilde{\phi}_a, (x_0, z)) < -\epsilon_0$ then $\Sigma(x_0, z) \not\models \phi$.*

Proof. To show that the algorithm ends, first note that $|\rho(\phi, s) - \rho(\phi, s')| < \epsilon$ if $\|s - s'\|_\infty < \epsilon$ (for STL formulas with predicates of the form $s_i \sim \lambda$. For general Lipschitz continuous predicates, the result holds as well using a bound dependent on the Lipschitz constants of the predicates). The algorithm continues for as long as new samples are added (Lines 6 and 10). Every new UNSAT (respectively SAT) sample has robustness more than ϵ with respect to the current $\tilde{\phi}_a$ (respectively $\neg\tilde{\phi}_a$), while all current UNSAT (respectively SAT) samples have negative robustness. Then, the distance (in infinite norm) of the new sample must be more than ϵ from other UNSAT (respectively SAT) samples. Since the space is bounded (Assumption 4.1), the process must end.

Since the algorithm ends, the last loop must be with $\epsilon = \epsilon_0$ and Line 12 is reached. Then, ρ in Line 4 is positive, i.e., $\forall(x_0, z) : \rho(\tilde{\phi}_a, (x_0, z)) > \epsilon_0 \implies \rho(\phi, \Sigma(x_0, z)) > 0$, and ρ in Line 8 is negative, i.e., $\forall(x_0, z) : \rho(\tilde{\phi}_a, (x_0, z)) < -\epsilon_0 \implies \rho(\phi, \Sigma(x_0, z)) < 0$. \square

4.4 Case Study

We illustrate the assumption mining algorithm on a 2D first order system with no external inputs so that the results can be visualized.

Let $\Sigma_{ex}(x_0, z)$ be the system described by the following matrices:

$$A = \begin{pmatrix} 0.3 & -0.1 \\ 0.1 & 0.3 \end{pmatrix}, D = 0, F = 0, \quad (4.5)$$

with $\Omega = [-10, 10] \times [-10, 10]$. The STL specification is the following:

$$\phi_{ex} = (\mathbf{G}_{[2,4]}x_0 > -4) \vee (\mathbf{G}_{[2,4]}x_0 < 4). \quad (4.6)$$

Algorithm 4: Assumption Mining

Input: Σ – System, ϕ – STL formula over x , ϵ_0 – target tolerance, ϵ – initial tolerance, α – learning rate,
Output: $\tilde{\phi}_a$ – STL formula over (x_0, z)

```

1 samples  $\leftarrow \emptyset$ 
2 while  $\epsilon \geq \epsilon_0$  do
3    $\tilde{\phi}_a \leftarrow \text{Fit}(\textit{samples})$ 
4    $\rho \leftarrow \min_{x_0, z} \rho(\phi, \Sigma(x_0, z))$  s.t.  $\Sigma(x_0, z) \models \phi; \rho(\tilde{\phi}_a, (x_0, z)) > \epsilon$ 
5   if  $\rho < 0$  then
6      $\textit{samples} \leftarrow \textit{samples} \cup ((\bar{x}_0, \bar{z}), \textit{UNSAT})$ 
7   else
8      $\rho \leftarrow \max_{x_0, z} \rho(\phi, \Sigma(x_0, z))$  s.t.  $\Sigma(x_0, z) \models \phi; \rho(\neg\tilde{\phi}_a, (x_0, z)) > \epsilon$ 
9     if  $\rho > 0$  then
10       $\textit{samples} \leftarrow \textit{samples} \cup ((\bar{x}_0, \bar{z}), \textit{SAT})$ 
11     else
12       $\epsilon \leftarrow \alpha\epsilon$ 
13 return  $\tilde{\phi}_a$ 

```

We ran the assumption mining algorithm with a target tolerance $\epsilon_0 = 0.75$, initial tolerance $\epsilon = 1.5$ and learning rate $\alpha = 0.5$. It took 150 seconds and 88 samples to produce the approximate assumption STL formula $\tilde{\phi}_a$ represented in Fig. 4-1 as the blue set.

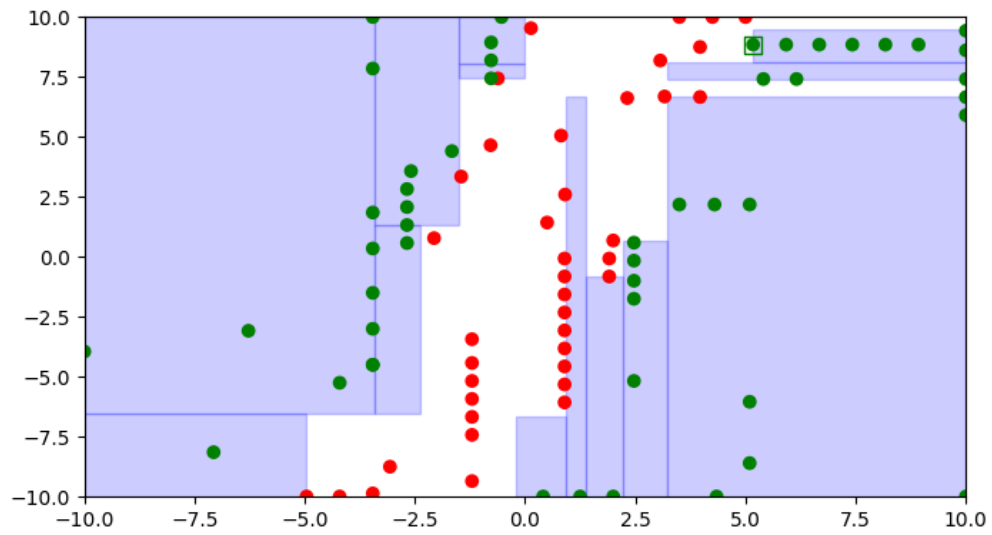


Figure 4.1: Final iteration before the target tolerance is reached. Shown in green are initial states contained in the assumption set, while those in red are outside of it. We show in blue the set of states that satisfy $\tilde{\phi}_a$.

Chapter 5

Constitutive Property Design

Our main goal in this chapter is to produce a wide variety of mechanical behaviors out of a single material. For example, can we give an elastic material a tunable, effective yield point, or make it strain harden or soften at a specified amount of displacement? This response to loading can be determined from the stress-strain curve, which serves as a fundamental piece of information that describes the global mechanical properties of structures and materials. Recent advances in mechanical metamaterials have shown that it is possible to tailor the relationship between stress and strain of a particular material through structural patterning (Bertoldi et al., 2008).

In order to formally specify the desired response of the structure, we introduce a formal language over stress-strain curves as well as show how S-STL can be adapted for this purpose. Then, we propose an optimization procedure based on gradient-free optimization algorithms and simulation. We illustrate the viability of our approach with an example featuring elastomeric structures.

5.1 Stress-Strain Curves and Logics

We first describe a model for the stress-strain response of a structure, then we introduce a new logic capable of describing properties of interest of the response of the structure. This logic, which we call Stress-Strain Predicate Logic (SSPL) is a predicate logic that captures most quantities of interest in a stress-strain curve in a user-friendly fashion. Alternatively, the spatio-temporal logic described in Chapter 2,

S-STL, can be easily adapted to this problem.

A stress-strain (s-s) curve $\sigma(\epsilon)$ is a vector valued function $\sigma(\epsilon) = (\sigma^+(\epsilon), \sigma^-(\epsilon))$, where $\sigma^+(\epsilon)$ represents the relationship between stress and strain in a material when a load is placed upon it (increasing strain), while $\sigma^-(\epsilon)$ is produced when the load is reduced (decreasing strain). There are well-known methods used to obtain a s-s curve from a specimen of the material or through simulation. We require the curves to satisfy the following natural assumptions: 1) $\sigma^+(0) = \sigma^-(0) = 0$; 2) $\sigma^+(\epsilon_u) = \sigma^-(\epsilon_u)$, $\epsilon_u = \max(\text{Domain}(\sigma))$; 3) $\sigma^+(\epsilon) \geq \sigma^-(\epsilon)$, $\forall \epsilon$.

A formula ϕ in SSPL is described as a conjunction of predicates $\mu = (p, \alpha, I)$, where:

- $p \in \mathbb{R}$ is a priority,
- α is a real valued function of a s-s curve that produces a quantity of interest from the curve, and
- $I = [a, b] \subset \mathbb{R}$ is an interval that represents the allowed or desired range of the quantity produced by α .

A curve σ satisfies $\mu = (p, \alpha, I)$ iff $\alpha(\sigma) \in I$. As an example, consider the following real language specification: “The positive Young’s Modulus of the curve should be within 100 GPa and 125 GPa with high priority and the positive yield stress should be within 200 MPa and 250 MPa with normal priority”. The corresponding formula would be:

$$\phi_{example} = (2.0, E_1^+, [100 \cdot 10^9, 125 \cdot 10^9]) \wedge (1.0, \sigma_y^+, [200 \cdot 10^6, 250 \cdot 10^6]), \quad (5.1)$$

where E_1^+ computes the Young’s Modulus of the positive curve (σ^+) and σ_y^+ computes its yield stress. Some quantities of interest are defined next and represented graphically over a typical s-s curve in Fig. 5.1. A user of this language will usually

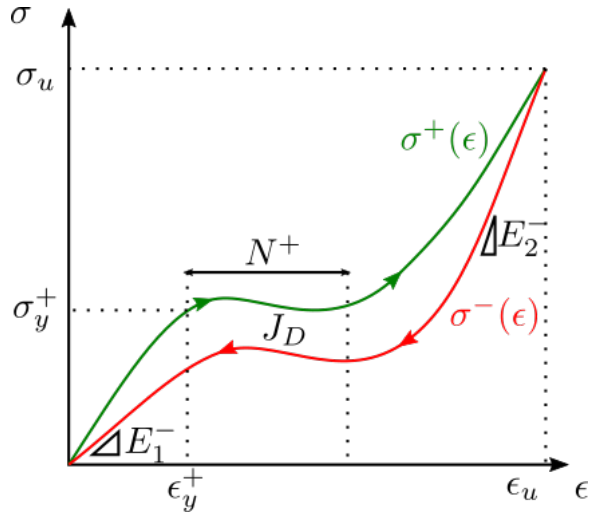


Figure 5.1: Typical s-s curve

be presented with a collection of quantities of interest to be used to define a specification. Additionally, the user can define new quantities of interest as needed. In the following, $o \in \{+, -\}$.

- ϵ_u : the maximum strain reached by the curve.
- σ_u : the stress corresponding to the maximum stress reached by the curve.
- E_1^o : the slope of the curve at strain close to 0 (increasing or decreasing).
- E_2^o : the slope of the curve at strain close to maximum.
- ϵ_y^o : the yield strain at which the curve first stops exhibiting linear behavior when increasing strain from 0 (correspondingly, when it becomes linear until 0 strain when decreasing it).
- σ_y^o : the yield stress corresponding to the yield strain.
- J_D : the area between the increasing and decreasing parts of the s-s curve.
- N^o : the necking region, i.e., the amount of additional strain one can apply to the material after the yielding point while keeping the stress almost level.

For some applications, knowing whether a s-s curve satisfies or not the specification is not enough and a numerical score related to how much the specification is violated or how robustly it is satisfied is needed. We call these scores the quantitative semantics of the language. We propose two kind of scores: an STL-like score extended with priorities and a smooth score.

STL-like: We define the robustness of a predicate, r_{STL} , as in STL without time, i.e., $r_{STL}(\mu, \sigma) = \min\{\alpha(\sigma) - a, b - \alpha(\sigma)\}$ and the usual rule for conjunctions $r_{STL}(\mu_1 \wedge \mu_2, \sigma) = \min\{r_{STL}(\mu_1, \sigma), r_{STL}(\mu_2, \sigma)\}$. From this robustness we define the score as the degree of violation of the formula weighted by the priorities of each predicate:

$$r(\bigwedge_i \mu_i, \sigma) = \sum_{i|\sigma \not\models \mu_i} p_i r_{STL}(\mu_i, \sigma). \quad (5.2)$$

Note that $r(\phi, \sigma) < 0 \iff \sigma \not\models \phi$. Since no quantitative information is obtained for satisfied formulas, we can switch to r_{STL} when $r = 0$ at the cost of ignoring priorities in that case.

Smooth score: We use three steps to define a smooth function r such that $r(\phi, \sigma) > 0 \iff \sigma \models \phi$.

- The basic score for a predicate μ is computed as $s(\mu, \sigma) = \frac{\alpha(\sigma) - a}{b - a}$. Note that $\sigma \models \mu \iff s(\mu, \sigma) \in [0, 1]$ and the best score should be assigned to $s = 0.5$.
- We transform s in the following way: when $s(\mu, \sigma) \notin [0, 1]$ it should be 0. Inside $[0, 1]$, it should increase smoothly from 0 until a maximum of 1 at $s = 0.5$ then decrease smoothly back to 0. We call this transformation T . There are several options for T such as appropriately scaled sin and many sigmoid-like functions.
- We define r using exponentiation for priorities and product for conjunctions:

$$r(\bigwedge_i \mu_i, \sigma) = \prod_i T(s(\mu_i, \sigma))^{p_i}. \quad (5.3)$$

We show in Fig. 5-2 how the score for a single predicate would look like with different priorities.

This score has the opposite problem from the STL-like score: we lose all quantitative information when the specification is violated. We can solve this issue by switching to the STL-like score when $r = 0$. Alternatively, in some applications where these scores are used to guide a search for the best satisfying s-s curve, the search can be started with a relaxed smooth score by increasing the size of the predicate intervals. Once a s-s curve is found that satisfies the relaxed formula, the relaxation can be reduced.

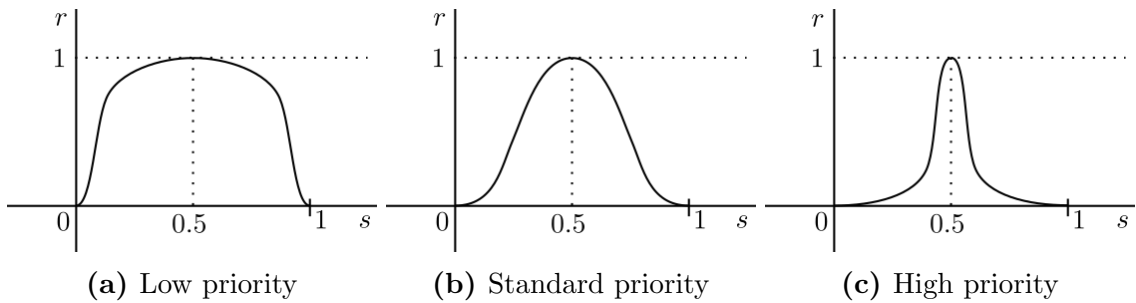


Figure 5-2: Shape of the smooth score against the basic score of a predicate with different priorities.

We show in Fig. 5-3 how the typical user would define a specification using the

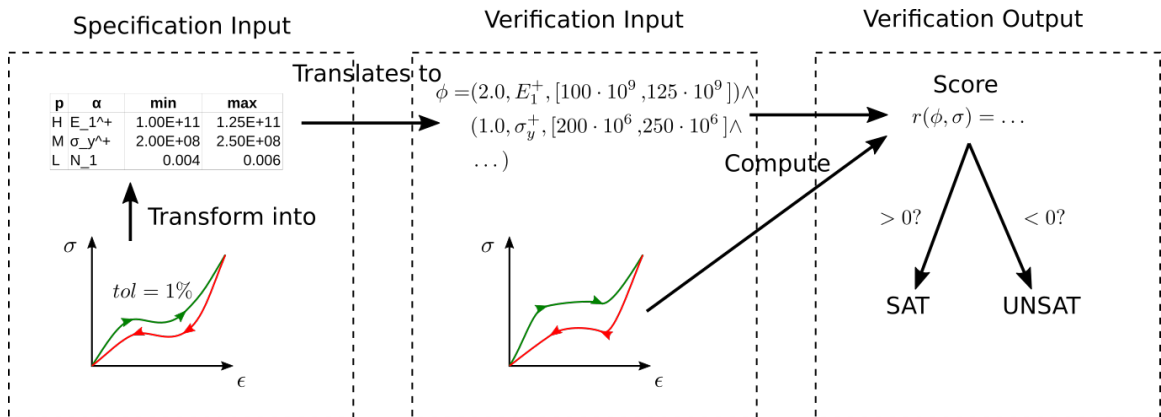


Figure 5-3: Proposed workflow for Stress-Strain curve verification.

language defined above and use it to verify whether a given s-s curve satisfies it. First, the user would have a choice of drawing the desired curve and supplying an allowed tolerance (measured with respect to the quantities of interest), or inputting directly the priority and allowed interval for each quantity of interest available in a predefined collection. If a drawing is supplied, it is then automatically converted to the direct input which can be then inspected and modified by the user. From this input, the formula ϕ is automatically constructed. When the user wants to verify whether a given s-s curve σ satisfies the specification, the score of the formula ϕ is computed for σ using the scores defined above. The score is shown to the user, who can infer the satisfaction of the formula from the sign of the score (positive iff satisfies) and by how much the formula is satisfied or violated (with high positive values meaning very robustly satisfying and low negative values meaning very far from satisfaction).

5.2 Problem Formulation and Solution

Beyond verification, we are interested in the synthesis of a structural pattern that allows a material to exhibit the specified constitutive response. The problem formulation has the following form:

Problem 5.1 (Constitutive Response Design). *Given a specification over s-s curves ϕ , a block of material, and constraints on the patterning allowed, find an allowed patterned structure with a stress-strain relationship that satisfies ϕ .*

We propose to solve Problem 5.1 by optimizing the score of the formula with respect to s-s curves obtained by simulation of the patterned structures. This optimization is done using a gradient-free optimization algorithm.

Given a desired constitutive response of the material formally specified as a formula ϕ in either SSPL or S-STL, an optimization process is used to synthesize a geometric configuration of the material that produces a satisfying constitutive re-

sponse. Formally, the optimization problem that we solve is the following:

$$\begin{aligned}
 & \max_p \quad r(\phi, \sigma(\cdot; p)) \\
 & \text{s.t.} \quad p \in \mathcal{P}, \\
 & \quad \quad \text{Valid}(p),
 \end{aligned} \tag{5.4}$$

where r is an appropriate score for the language, \mathcal{P} is the parameter space that describes the patterns, Valid are general constraints on the patterns given by the user and $\sigma(\cdot; p)$ is the s-s curve of the material produced using the pattern given by p .

Regardless of the choice of language, score or parameterization, our optimization process has the following basic structure: first, we produce a parameterized model of the patterned material in a simulation tool such as ANSYS or ABAQUS. Then, we configure the simulation tool so that a s-s curve can be obtained for any valid pattern with enough accuracy and speed. Once this is set up, we can begin the optimization itself, which is non-linear, non-convex and possibly the objective function is not even continuous with respect to the parameters. Thus, a general gradient-free optimization method such as differential evolution (Storn and Price, 1997) must be used. Each parameter is evaluated by simulating the corresponding patterned material, obtaining the s-s curve and computing its score with respect to the specification.

5.3 Case Study

We present here an example highlighting the solution for the synthesis problem based on the work presented in (Shim et al., 2013b).

A square sheet of rubber of side $L = 8$ cm and thickness $h = 1$ cm is drilled with circular holes of radius r constrained to be within the interval $[0.3 \text{ cm}, 0.4 \text{ cm}]$ and distributed in a lattice given by the vectors $v_1, v_2 \in \mathbb{R}^2$ such that they do not

intersect. The constitutive response was obtained as a positive force-displacement curve (analogous to the positive part of an s-s curve σ^+) by loading the material with a force $F(t)$ applied uniformly across the top surface such that the displacement at the top surface increases linearly from 0 to 1.05 cm in 3.5 seconds. The specification in this case is to have the force-displacement curve of the material to be within ϵ tolerance of a target curve p , which can be expressed in S-STL as the following formula:

$$\phi = \forall x \in \{(0, L)\} : \mathbf{G}_{[0,T]}(F < p(d_2(x)) + \epsilon \wedge F > p(d_2(x)) - \epsilon),$$

$$p(d) = \begin{cases} \frac{40}{7.5 \cdot 10^{-3}} d & d < 7.5 \cdot 10^{-3}, \\ 40 & d > 7.5 \cdot 10^{-3}, \end{cases} \quad (5.5)$$

where $d_2(x)$ represents the displacement of the top boundary at x , F is the applied force, the temporal operator $\mathbf{G}_{[0,T]}\psi$ means that ψ must be satisfied at all times in the interval and the spatial operator $\forall x \in \{(0, L)\} : \psi$ means that ψ must be satisfied at all points in the interval. Note that, compared SSPL, this logic has two main disadvantages: it explicitly considers time in an essentially static problem; and it is not well suited to the description of s-s curves, as it does not allow explicit references to the positive and negative parts of the curve, tolerance is considered with respect to the full curve, and quantities of interest of the curve cannot be directly referenced with ease.

We solve this problem using differential evolution to find the best pattern with respect to the S-STL score of the specification ϕ . We generate simulations of the system using the FEM simulator ANSYS. After 280 evaluations performed in about 38 hours, we obtained the pattern depicted in Fig. 5.4a, which results in a S-STL score of 0.641. We show in Fig. 5.4c the resulting force-displacement curve as well as the satisfying region for ϕ . It is important to note that these FEM simulations were

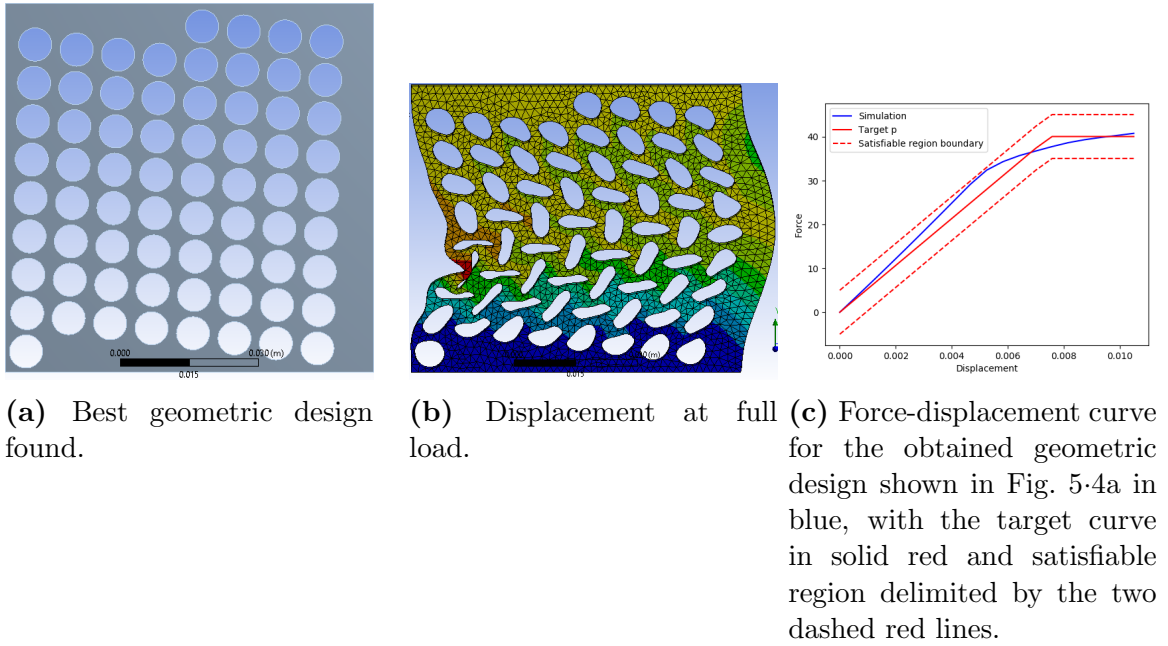


Figure 5-4: Results for constitutive design.

run accounting for both geometric nonlinearity (i.e., finite deformation kinematics) as well as material nonlinearity (a Neo-Hookean material model for rubber elasticity).

Chapter 6

Conclusions and Future Work

In this dissertation we considered a formal methods approach to the design of PDE systems. In particular, we focused on two design problems: a boundary control problem with spatio-temporal specifications over the state of the PDE, and a constitutive response design problem where a geometric structure must be obtained for a specification over the intrinsic properties of the PDE.

In the first part of the dissertation, we formulated and solved boundary control synthesis and verification problems for systems governed by a multi-dimensional, possibly non-linear, PDE with specifications given in an extension to STL. Our solution relies on the approximation of the PDE using the FEM, which reduces the problem to the control synthesis of a discrete-time linear system under regular STL constraints. The reformulation requires correcting the predicates in the formula using the FEM approximation errors, as well as the derivatives of the approximated solution and the predicate functions, to account for approximation and discretization errors. Finally, the resulting control problem is encoded as a MILP, which we show can be solved in minutes when a good approximation to the optimal solution obtained using gradient-free optimization methods can be used as a starting point for the MILP solver. However, our method becomes untractable as the complexity of the PDE increases. We explored the performance of the method in 1D, 2D, first order, second order, linear and nonlinear PDEs. Plans for future work include studying the viability of the approach on more complex nonlinear PDEs.

In the second part of the dissertation, we proposed a sampled based algorithm for assumption mining in non-monotone systems. We follow existing work by iteratively constructing an approximation to the assumption set obtaining samples both within and without the set. While in monotone systems this sets have a structure that makes them easy to define, we instead represent them as STL formulas inferred from the set of samples. As future work, we plan to apply the assumption mining algorithm in more challenging scenarios. In addition, we plan to build on top of this algorithm towards a full decentralized control synthesis framework based on assume-guarantee contracts.

Also in this part, we presented an STL inference framework based on decision trees. The proposed framework describes decision-tree learning algorithms which may be customized through a set of primitive properties of interest, an impurity measure which captures the node’s homogeneity, and stopping conditions for the algorithm. The performance advantage of the proposed procedures is due to the incremental nature of growing STL formulas represented as trees. We also defined extended versions of the classical impurity measures such that these take into account the robustness degrees of signals. We argue that the extended versions of the impurity measures increase the generalization capability of the resulting formulas.

In the final part of the dissertation, we considered the problem of designing the geometry of structures such that a specification over the constitutive response of the structure is met. We defined a predicate logic capable of describing the constitutive response of a mechanical system in a user-friendly way, although we also showed how S-STL can be used as well. Then, we described an optimization procedure that can be used to generate the geometric design that produces the best fitting constitutive response. We showed the viability of our approach on elastomeric structures. In our future work, we plan to implement the proposed SSPL logic and explore the

different proposed scores. Moreover, we plan to improve the optimization procedure. As the model simulation is very expensive, a possible improvement would be to cache simulation results. Since these results contain a very large amount of data, a machine learning technique could be used to obtain an approximate map from the model parameters to the minimal amount of data required to compute the score.

6.1 Future Research Directions

From this dissertation, we foresee two main avenues of future research. The first one focuses on improving the framework proposed in the first part of the dissertation as a solution to the tunable fields problem. In its current state, the range of problems we can solve is limited to small PDE models with simple geometries that can be accurately approximated using very simple FEM models with a small number of elements. However, most real world problems require FEM models with several orders of magnitude more elements. We consider our framework to have the potential to solve such problems, at least in the linear case, but a scalable solution must first be developed. In the second part of the dissertation we started work on one such solution based on assume-guarantee contracts. In addition, the development of this solution or others to the scalability problem is, in itself, an interesting and challenging problem in the field of formal methods with applications in other fields such as spatially distributed systems.

The second main research direction is in the solution of the constitutive design problem. Our proposed solution can be improved in several ways, such as by further exploring the proposed SSPL logic or by designing a better optimization procedure that takes into account the high cost of model simulation. Once the framework is refined to the point where it can handle real world models and specifications that can be put to test in the laboratory, collaboration with materials science researchers can

yield results of interest not only to the formal methods community but also in the materials science community.

Besides these two main directions, we should also mention the following idea: would it be possible to combine the tunable fields and constitutive response design problems? Could we develop a formal methods framework that would allow us to design a structure with a desired intrinsic behavior as well as boundary conditions that operate the structure such that it satisfies a given specification? Note that the difficulty here is that for some allowed constitutive responses, satisfaction of the field specification might prove unfeasible. At present, we consider this combined problem the ultimate goal of research in formal methods for PDEs.

References

- Abbas, H., Mittelman, H., and Fainekos, G. (2014). Formal property verification in a conformance testing framework. In *2014 Twelfth ACM/IEEE Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 155–164. IEEE.
- Alberdi, R. and Khandelwal, K. (2019). Bi-material topology optimization for energy dissipation with inertia and material rate effects under finite deformations. *Finite Elements in Analysis and Design*, 164:18–41.
- Allaire, G., Jouve, F., and Toader, A.-M. (2004). Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics*, 194:363–393.
- Asarin, E., Donzé, A., Maler, O., and Nickovic, D. (2012). Parametric identification of temporal properties. In *Runtime Verification*, pages 147–160. Springer.
- Bartocci, E., Gol, E. A., Haghghi, I., and Belta, C. (2016). A Formal Methods Approach to Pattern Recognition and Synthesis in Reaction Diffusion Networks. *IEEE Transactions on Control of Network Systems*, PP(99):1–1.
- Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427.
- Bendsoe, M. P. (1989). Optimal shape design as a material distribution problem. *Structural Optimization*, 1:193–202.
- Bendsoe, M. P. and Sigmund, O. (1999). Material interpolation schemes in topology optimization. *Archives of Applied Mechanics*, 69:635–654.
- Bertoldi, K., C. Boyce, M., Deschanel, S., M. Prange, S., and Mullin, T. (2008). Mechanics of deformation-triggered pattern transformations and superelastic behavior in periodic elastomeric structures. *Journal of The Mechanics and Physics of Solids*, 56:2642–2668.
- Bertoldi, K., Vitelli, V., Christensen, J., and van Hecke, M. (2017). Flexible mechanical metamaterials. *Nature Reviews Materials*, 2:17066.
- Bombara, G. (2020). *Learning Temporal Logic Formulae from Data*. PhD thesis, Boston University (<https://open.bu.edu/handle/2144/39359>).

- Bombara, G., Vasile, C.-I., Penedo, F., Yasuoka, H., and Belta, C. (2016). A Decision Tree Approach to Data Classification Using Signal Temporal Logic. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 1–10, New York, NY, USA. ACM.
- Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.
- Bruns, T. E. and Tortorelli, D. A. (2001). Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer Methods in Applied Mechanics and Engineering*, 190:3443–3459.
- Bruns, T. E. and Tortorelli, D. A. (2003). An element removal and reintroduction strategy for the topology optimization of structures and compliant mechanisms. *International Journal for Numerical Methods in Engineering*, 57:1413–1430.
- Buhl, T., Pedersen, C. B. W., and Sigmund, O. (2000). Stiffness design of geometrically nonlinear structures using topology optimization. *Structural and Multidisciplinary Optimization*, 19:93–104.
- Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3):15:1–15:58.
- Clausen, A., Wang, F., Jensen, J. S., Sigmund, O., and Lewis, J. A. (2015). Topology optimized architectures with programmable poisson’s ratio over large deformations. *Advanced Materials*, 27:5523–5527.
- Cormen, T. H. (2009). *Introduction to Algorithms*. MIT Press, third edition.
- Deaton, J. D. and Grandhi, R. V. (2014). A survey of structural and multidisciplinary continuum topology optimization: post 2000. *Structural and Multidisciplinary Optimization*, 49:1–38.
- Donzé, A., Ferrere, T., and Maler, O. (2013). Efficient robust monitoring for STL. In *Computer Aided Verification*, pages 264–279. Springer.
- Donzé, A. and Maler, O. (2010). Robust Satisfaction of Temporal Logic over Real-Valued Signals. In *Formal Modeling and Analysis of Timed Systems. FORMATS 2010. Lecture Notes in Computer Science*, number 6246, pages 92–106. Springer Berlin Heidelberg.
- Ezio Bartocci, Luca Bortolussi, Michele Loreti, and Laura Nenzi (2017). Monitoring Mobile and Spatially Distributed Cyber-Physical Systems. In *2017 ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*.

- Fainekos, G. E. and Pappas, G. J. (2009). Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291.
- Filipov, E. T., Chun, J., Paulino, G. H., and Song, J. (2016). Polygonal multiresolution topology optimization (polymtop) for structural dynamics. *Structural and Multidisciplinary Optimization*, 53:673–694.
- Frei, W. R., Tortorelli, D. A., and Johnson, H. T. (2005). Topology optimization of a photonic crystal waveguide termination to maximize directional emission. *Applied Physics Letters*, 86:111114.
- Gea, H. C. and Luo, J. (2001). Topology optimization of structures with geometrical nonlinearities. *Computers and Structures*, 79:1977–1985.
- Gerth, R., Peled, D., Vardi, M. Y., and Wolper, P. (1996). Simple On-the-fly Automatic Verification of Linear Temporal Logic. In *Protocol Specification, Testing and Verification XV*, pages 3–18. Springer US.
- Ghasemi, H., Park, H. S., and Rabczuk, T. (2017). A level-set based iga formulation for topology optimization of flexoelectric materials. *Computer Methods in Applied Mechanics and Engineering*, 313:239–258.
- Grosu, R., Smolka, S. A., Corradini, F., Wasilewska, A., Entcheva, E., and Bartocci, E. (2009). Learning and Detecting Emergent Behavior in Networks of Cardiac Myocytes. *Communications of the ACM*, 52(3):97–105.
- Gurobi Optimization, I. (2016). Gurobi Optimizer Reference Manual. *URL: <http://www.gurobi.com>*.
- Haghighi, I., Jones, A., Kong, Z., Bartocci, E., Gros, R., and Belta, C. (2015). SpaTeL: A Novel Spatial-temporal Logic and Its Applications to Networked Systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 189–198, New York, NY, USA. ACM.
- Hoxha, B., Abbas, H., and Fainekos, G. E. (2014). Benchmarks for Temporal Logic Requirements for Automotive Systems. In *Goran Frehse and Matthias Althoff (editors). ARCH14-15. 1st and 2nd International Workshop on Applied verification for Continuous and Hybrid Systems*, volume 34, pages 25–30.
- Hughes, T. J. R. (2000). *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, Mineola, NY.
- Hyafil, L. and Rivest, R. L. (1976). Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17.
- Isermann, R. (2006). *Fault-diagnosis systems*. Springer.

- Jin, X., Donzé, A., Deshmukh, J. V., and Seshia, S. A. (2015). Mining Requirements From Closed-Loop Control Models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(11):1704–1717.
- Jones, A., Kong, Z., and Belta, C. (2014). Anomaly detection in cyber-physical systems: A formal methods approach. In *2014 IEEE 53rd Annual Conference on Decision and Control (CDC)*, pages 848–853. IEEE.
- Jung, D. and Gea, H. C. (2004). Topology optimization of nonlinear structures. *Finite Elements in Analysis and Design*, 40:1417–1427.
- Kim, E. S., Arcak, M., and Seshia, S. A. (2016). Directed Specifications and Assumption Mining for Monotone Dynamical Systems. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 21–30, New York, NY, USA. ACM.
- Kloetzer, M. and Belta, C. (2008). A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297.
- Kochmann, D. M. and Bertoldi, K. (2017). Exploiting microstructural instabilities in solids and structures: from metamaterials to structural transitions. *Applied Mechanics Reviews*, 69:050801.
- Kong, Z., Jones, A., and Belta, C. (2017). Temporal Logics for Learning and Detection of Anomalous Behaviors. *IEEE Transactions on Automatic Control*, 62:1210–1222.
- Kong, Z., Jones, A., Medina Ayala, A., Aydin Gol, E., and Belta, C. (2014). Temporal Logic Inference for Classification and Prediction from Data. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control, HSCC '14*, pages 273–282, New York, NY, USA. ACM.
- Kowalska, K. and Peel, L. (2012). Maritime anomaly detection using Gaussian Process active learning. In *2012 15th International Conference on Information Fusion (FUSION)*, pages 1164–1171.
- Krstic, M. and Smyshlyaev, A. (2008). *Boundary Control of PDEs: A Course on Backstepping Designs*. Siam.
- Lavan, O. (2019). Adjoint sensitivity analysis and optimization of transient problems using the mixed lagrangian formalism as a time integration scheme. *Structural and Multidisciplinary Optimization*. in press.
- Li, W., Dworkin, L., and Seshia, S. A. (2011). Mining assumptions for synthesis. In *Ninth ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMPCODE2011)*, pages 43–50.

- Maler, O. and Nickovic, D. (2004). Monitoring Temporal Properties of Continuous Signals. In Lakhnech, Y. and Yovine, S., editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, number 3253 in Lecture Notes in Computer Science, pages 152–166. Springer Berlin Heidelberg.
- Maute, K., Schwarz, S., and Ramm, E. (1998). Adaptive topology optimization of elastoplastic structures. *Structural Optimization*, 15:81–91.
- Maute, K., Tkachuk, A., Wu, J., Qi, H. J., Deng, Z., and Dunn, M. L. (2015). Level set topology optimization of printed active composites. *Journal of Mechanical Design*, 137:111402.
- Meurer, T. and Kugi, A. (2009). Tracking control for boundary controlled parabolic PDEs with varying parameters: Combining backstepping and differential flatness. *Automatica*, 45(5):1182–1194.
- Nakshatrala, P. B. and Tortorelli, D. A. (2015). Topology optimization for effective energy propagation in rate-independent elastoplastic material systems. *Computer Methods in Applied Mechanics and Engineering*, 295:305–326.
- Nanthakumar, S. S., Lahmer, T., Zhuang, X., Park, H. S., and Rabczuk, T. (2016). Topology optimization of piezoelectric nanostructures. *Journal of the Mechanics and Physics of Solids*, 94:316–335.
- Nanthakumar, S. S., Valizadeh, N., Park, H. S., and Rabczuk, T. (2015). Surface effects on shape and topology optimization of nanostructures. *Computational Mechanics*, 56:97–112.
- Nanthakumar, S. S., Zhuang, X., Park, H. S., and Rabczuk, T. (2017). Topology optimization of flexoelectric structures. *Journal of the Mechanics and Physics of Solids*, 105:217–234.
- Osanov, M. and Guest, J. K. (2016). Topology optimization for architected materials design. *Annual Review of Materials Research*, 46:211–233.
- Penedo, F., Park, H., and Belta, C. (2018). Control Synthesis for Partial Differential Equations from Spatio-Temporal Specifications. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4890–4895.
- Picelli, R., Townsend, S., Brampton, C., Norato, J., and Kim, H. A. (2018). Stress-based shape and topology optimization with the level set method. *Computer Methods in Applied Mechanics and Engineering*, 329:1–23.
- Quinlan, J. R. (2014). *C4.5: Programs for Machine Learning*. Elsevier.

- Raman, V., Donzé, A., Maasoumy, M., Murray, R. M., Sangiovanni-Vincentelli, A., and Seshia, S. A. (2014). Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 81–87.
- Reis, P. M., Jaeger, H. M., and van Hecke, M. (2015). Designer matter: a perspective. *Extreme Mechanics Letters*, 5:25–29.
- Ripley, B. D. (1996). *Pattern recognition and neural networks*. Cambridge university press.
- Sadraddini, S. and Belta, C. (2015). Robust temporal logic model predictive control. In *2015 53rd Annual Allerton Conference On Communication, Control, and Computing (Allerton)*, pages 772–779. IEEE.
- Sadraddini, S., Rudan, J., and Belta, C. (2017). Formal Synthesis of Distributed Optimal Traffic Control Policies. In *Proceedings of the 8th International Conference on Cyber-Physical Systems, ICCPS '17*, New York, NY, USA. ACM.
- Shim, J., Shan, S., Košmrlj, A., Kang, S. H., Chen, E. R., Weaver, J. C., and Bertoldi, K. (2013a). Harnessing instabilities for design of soft reconfigurable auxetic/chiral materials. *Soft Matter*, 9(34):8198–8202.
- Shim, J., Shan, S., Kosmrlj, A., Kang, S. H., Chen, E. R., Weaver, J. C., and Bertoldi, K. (2013b). Harnessing instabilities for design of soft reconfigurable auxetic/chiral materials. *Soft Matter*, 9:8198–8202.
- Sigmund, O. and Maute, K. (2013). Topology optimization approaches: a comparative review. *Structural and Multidisciplinary Optimization*, 48:1031–1055.
- Sigmund, O. and Torquato, S. (1999). Design of smart composite materials using topology optimization. *Smart Structures and Materials*, 8:365–379.
- Storn, R. and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359.
- Swan, C. C. and Kosaka, I. (1997). Voigt-reuss topology optimization for structures with nonlinear material behavior. *International Journal for Numerical Methods in Engineering*, 40:3785–3814.
- van Dijk, N. P., Maute, K., Langelaar, M., and van Keulen, F. (2013). Level-set methods for structural topology optimization: a review. *Structural and Multidisciplinary Optimization*, 48:437–472.

- Wang, F., Lazarov, B. S., Sigmund, O., and Jensen, J. S. (2014). Interpolation scheme for fictitious domain techniques and topology optimization of finite strain elastic problems. *Computer Methods in Applied Mechanics and Engineering*, 276:453–472.
- Wang, M. Y., Wang, X., and Guo, D. (2003). A level set method for structural topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 192:227–246.
- Yang, H., Hoxha, B., and Fainekos, G. (2012). Querying Parametric Temporal Logic Properties on Embedded Systems. In *Testing Software and Systems*, number 7641 in Lecture Notes in Computer Science, pages 136–151. Springer.
- Yu, X., Zhou, J., Liang, H., Jiang, Z., and Wu, L. (2018). Mechanical metamaterials associated with stiffness, rigidity and compressibility: a brief review. *Progress in Materials Science*, 94:114–173.
- Zhang, X., Jr, A. S. R., and Paulino, G. H. (2017). Material nonlinear topology optimization using the ground structure method with a discrete filtering scheme. *Structural and Multidisciplinary Optimization*, 55:2045–2072.
- Zuliani, P., Platzer, A., and Clarke, E. M. (2013). Bayesian statistical model checking with application to Stateflow/Simulink verification. *Formal Methods in System Design*, 43(2):338–367.

CURRICULUM VITAE

