COMPOSITIONAL ANALYSIS OF NETWORKED CYBER-PHYSICAL SYSTEMS: SAFETY AND PRIVACY

BY

ZHENQI HUANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Associate Professor Sayan Mitra, Chair

Professor Geir E. Dullerud

Professor Marta Z. Kwiatkowska, University of Oxford

Professor Nitin H. Vaidya

ABSTRACT

Cyber-physical systems (CPS) are now commonplace in power grids, manufacturing, and embedded medical devices. Failures and attacks on these systems have caused significant social, environmental and financial losses. In this thesis, we develop techniques for proving invariance and privacy properties of cyber-physical systems that could aid the development of more robust and reliable systems.

The thesis uses three different modeling formalisms capturing different aspects of CPS. *Networked dynamical systems* are used for modeling (possibly time-delayed) interaction of ordinary differential equations, such as in power system and biological networks. *Labeled transition systems* are used for modeling discrete communications and updates, such as in sampled data-based control systems. Finally, *Markov chains* are used for describing distributed cyber-physical systems that rely on randomized algorithms for communication, such as in a crowd-sourced traffic monitoring and routing system. Despite the differences in these formalisms, any model of a CPS can be viewed as a mapping from a parameter space (for example, the set of initial states) to a space of behaviors (also called trajectories or executions). In each formalism, we define a notion of *sensitivity* that captures the change in trajectories as a function of the change in the parameters. We develop approaches for approximating these sensitivity functions, which in turn are used for analysis of invariance and privacy.

For proving invariance, we compute an over-approximation of *reach set*, which is the set of states visited by any trajectory. We introduce a notion of input-to-state (IS) discrepancy functions for components of large CPS, which roughly captures the sensitivity of the component to its initial state and input. We develop a method for constructing a reduced model of the entire system using the IS discrepancy functions. Then, we show that the trajectory of the reduced model over-approximates the sensitivity of the entire system

with respect to the initial states. Using the above results we develop a sound and relatively complete algorithm for compositional invariant verification.

In systems where distributed components take actions concurrently, there is a combinatorial explosion in the number of different action sequences (or traces). We develop a partial order reduction method for computing the reach set for these systems. Our approach uses the observation that some action pairs are *approximately independent*, such that executing these actions in any order results in states that are close to each other. Hence a (large) set of traces can be partitioned into a (small) set of equivalent classes, where equivalent traces are derived through swapping approximately independent action pairs. We quantify the sensitivity of the system with respect to swapping approximately independent action pairs, which upper-bounds the distance between executions with equivalent traces. Finally, we develop an algorithm for precisely over-approximating the reach set of these systems that only explore a reduced set of traces.

In many modern systems that allow users to share data, there exists a tension between improving the global performance and compromising user privacy. We propose a mechanism that guarantees ε-differential privacy for the participants, where each participant adds noise to its private data before sharing. The distributions of noise are specified by the sensitivity of the trajectory of agents to the private data. We analyze the trade-off between ε-differential privacy and performance, and show that the cost of differential privacy scales quadratically to the privacy level.

The thesis illustrates that quantitative bounds on sensitivity can be used for effective reachability analysis, partial order reduction, and in the design of privacy preserving distributed cyber-physical systems.

To my wife, parents and grandmother, for their endless love and support.

ACKNOWLEDGMENTS

This thesis would not be possible without the help from many others. I am truly fortunate to have had Prof. Sayan Mitra as my advisor. His patience, motivation, and knowledge kept inspiring me in all phases of my research: from identifying interesting topics, developing elegant solutions, to composing impressive presentations. Sayan is also a compassionate mentor who gave me moral support during difficult times.

I would like to thank the rest of my thesis committee: Prof. Geir Dullerud, Prof. Marta Kwiatkowska, and Prof. Nitin Vaidya for their insightful comments and generous guidance over the years. They shared their expertise in control theory, formal methods, distributed algorithms, and differential privacy with me. Their sharp questions helped me refining results, fixing mistakes, and identifying future directions.

I would like to thank Prof. Swarat Chudhuri, Prof. Marco Caccamo, Prof. Aranya Chakrabortty, Prof. Saman Zonouz, and many others, for providing advice and shaping my thought. My sincere thanks also goes to my academic peers, including Chuchu Fan, Yu Wang, Stanley Bak, Taylor Johnson, Sridhar Duggirala, Ritwika Ghosh, Hongxu Chen, Nicole Chan, Hussein Sibaie, Fardin Abdi, Yixiao Lin, Jeremy Green, and Karthik Manamcheri. We are good friends who are able to exchange ideas on and outside research.

I am grateful to my wife Iris, who truly understands me and literarily takes care of me. Your love completes me and makes me a better person. I am grateful to my mom, dad, and grandparents. They are always trying to provide me the best education and encourage me when I am feeling down. Finally, I would like to thank everyone else who has helped me.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

In modern engineering systems, computers interacting with physical processes have become commonplace. Many of these *cyber-physical systems* (CPS) consist of distributed components that send physical or cyber signals to each other through a network. It is natural to view these systems as networks of nodes and edges, where the nodes stand for computing units and the edges are communication channels (see Figure 1.1). The evolution of the state of a node over time is influenced by the states of its neighboring nodes.

Examples of such systems are abundant in automotive control systems, embedded medical devices, and in building control systems. For instance, a power network may be composed of nodes of generators, buses and loads. The state of each node captures local quantities such as phase angle, voltage and power consumption [1]. The evolution of these quantities is governed by the power-flow equations. Another example is a distributed air-traffic control system where each aircraft can be viewed as a node and the edges model pairwise communication channels [2]. The states of an aircraft capture its quantities such as position, velocity, attitude, which evolve according to aerodynamic and flight control protocols. The design of these systems should provide a high level of reliability as failures can cause huge financial, environmental and social losses. An important research challenge is to develop design and verification techniques for checking whether or not a given system meets certain desirable requirements. The properties of CPS we study in this thesis fall into the following broad categories.

Invariance. Invariance properties of the system are those that remain valid all the time. Roughly, these properties capture the notion that nothing "bad" can ever happen. For example, in the air traffic control system, aircraft should never violate the minimum separation requirements [3]. In a smart grid, the voltage should not exceed the design threshold [4]. Many safety

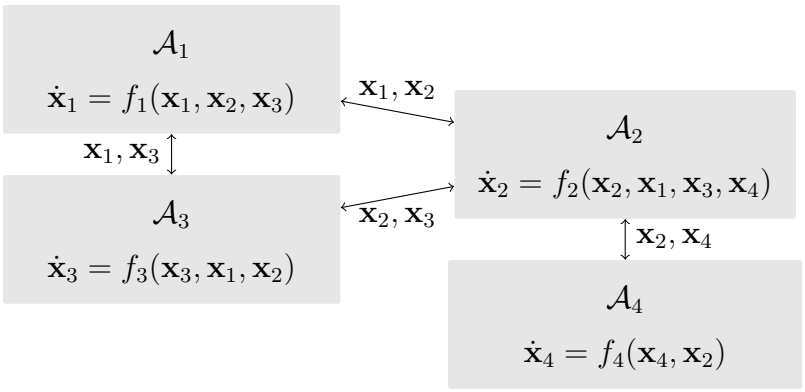


Figure 1.1: A network view of a cyber-physical system. The nodes are computing or physical components. Each node $\mathcal{A}_i$ has a state $\mathbf{x}_i$ that evolves overtime governed by difference equations or ordinary differential equations (ODE). An edge from node $\mathcal{A}_i$ to $\mathcal{A}_j$ indicates the state of $\mathbf{x}_i$ affects the dynamics of $\mathcal{A}_j$ and vice versa.

specifications of CPS are expressed as invariance properties.

Security & **Privacy.** Security and privacy cover a broad range of concerns for CPS. Roughly, a secure and private system should be able to protect information from unauthorized access, use, modification, inspection, recording, and destruction. One well-known security incident is the Stuxnet worm, which targeted industrial software used to control nuclear fuel processing plants [5]. The worm ultimately sabotaged and destroyed an Iranian facility by introducing malicious control inputs to actuators controlling uranium centrifuges. Other threats include violation of access rights [6], denial of service attacks [7], and de-anonymization [8]. In this thesis, we focus on privacy of users participating in a distributed system such as the navigation applications provided by Google maps and Waze [9]. In these systems, users share data (such as location) for better system-level performance (such as routing delay), but at the risk of compromising their privacy. The system is private if adversaries cannot infer the user's private data with high confidence, even with access to the communication of the system.

1.1 Formalisms

Cyber-physical systems can exhibit a broad range of behaviors due to the rich interactions between computers and physical environments. Several mathematical modeling frameworks for CPS have been proposed, such as hybrid automata [10, 11, 12], hybrid Petri nets [13, 14], switched systems [15], differential dynamic logics [16], hybrid dynamical systems [17] and hybrid process algebra [18]. In this thesis, we employ three formalisms that capture different salient aspects of CPS.

Networked Dynamical System (NDS) is a formalism where each node evolves according to ordinary differential equations (ODE) and an edge indicates that the linked nodes communicate with each other. The communication between nodes may experience time delays. A trajectory (or execution) of a networked dynamical system ξ_θ is the solution of the delayed differential equation with a given initial state θ. It is natural to model a large class of CPS as networked dynamical systems such as power grid [19], swarm robots [20], and embedded medical devices [21].

Labeled Transition System (LTS) is a standard state-machine model for computer programs [22, 23, 24]. The nodes take actions to communicate with each other and to update the local and shared states of the system. In our formalism, the nodes may nondeterministically choose actions to be taken. An execution of LTS ξ_τ is a sequence of states which are visited consecutively by taking a sequence of actions τ.

Markov Chain (MC) is a formalism where each transition is chosen according to a probability distribution. The Markov chain is a natural formalism to capture stochasticity in CPS arising from diverse sources such as noise, disturbance, stochastic dynamics and the coin-flip of algorithms. Specifically, we use Markov chains to model a class of distributed control systems, where participating agents cooperate to achieve local and global objectives.

1.2 Sensitivity Analysis

As mentioned in Section 1.1, we use different formalisms to capture salient features of different classes of cyber-physical systems. Any model in one of these formalisms can be viewed in the following abstract way. Roughly, a model specifies a mapping from a parameter domain to a space of *trajectories* (or executions). The parameter domain may correspond to a set of initial states, design choices, external inputs, or user data. Let $\xi_d(t)$ be a trajectory that is specified by some parameter value d. Roughly, the *sensitivity* of the system captures the change in the trajectories as a function of the change in the parameter, that is $|\xi_d(t) - \xi_{d'}(t)|$. For models with linear dynamics which have analytical solutions, sensitivity can be derived by directly comparing trajectories. For systems with nonlinear dynamics where explicit solutions are generally absent, sensitivity can be estimated by statistical testing [25] or numerical simulation [26].

In this thesis, we develop techniques to prove invariance and privacy properties of CPS, which both rely on computing bounds on the sensitivity of the models in question. We compute the bounds of sensitivity with respect to different parameters, and use them in two ways. In invariant verification, we compute the sensitivity with respect to initial states and action sequences. Our verification techniques involve generalizing a single trajectory to over-approximate a set of trajectories with similar initial states and action sequences. The bounds on the sensitivity determines how much generalization is provably sufficient. In contrast, for designing *differentially private* distributed systems, we compute the sensitivity with respect to the sensitive data of agents participating in the system. A differentially private mechanism ensures that adversaries with access to the messages in a network cannot infer the private data of individuals, such as initial state or control objectives, with high level of confidence. The design of a differentially private mechanism relies on adding noise to obscure the difference in the individuals' trajectories when any agent changes its private data. Hence, the distribution of noise depends on the sensitivity of agents' trajectory to the private data. We will summarize these approaches in the remainder of this introduction.

1.3 Invariant Verification with Input-to-State Discrepancy

The continuous evolution of states of CPS is usually modeled by ordinary differential equations (ODE). A trajectory of the model is specified by its initial state. Several recently developed approaches for invariant verification of these systems with nonlinear ODEs combine numerical simulations with static analysis to over-approximate the infinite number of behaviors of the system [27, 28, 29, 30, 31]. The common idea of these approaches is as follows. For a single initial state $\mathbf{x}$, let $\xi_{\mathbf{x}}$ be the numerically computed trajectory from $\mathbf{x}$ for a bounded time. Based on the continuous dependence of $\xi_{\mathbf{x}}$ on $\mathbf{x}$, we know that all the trajectories from neighboring initial states will be close to $\xi_{\mathbf{x}}$. With sensitivity analysis of the trajectories to the initial state, we get quantitative bounds on the distance between neighboring trajectories. This enables us to compute a tube around $\xi_{\mathbf{x}}$, that contains all possible trajectories from the neighborhood of $\mathbf{x}$. By repeating this process for different initial states, all reachable states from an initial set can be over-approximated and the invariance properties can be verified.

Generalizing several of these properties, in [29] the authors introduced the notion of *discrepancy function* as a continuous function (of the distance between initial states and time) characterizing the convergence or divergence rates of trajectories. It was shown that if a nonlinear, switched, or hybrid system model is annotated with appropriate discrepancy function(s) then the above approach of combining simulations and generalizations gives a sound and relatively complete algorithm for verifying bounded time invariants.

A challenge for these methods is to come up with a discrepancy function, which becomes increasingly difficult for larger models in which many components interact [32, 30, 31]. In this thesis, we address this problem by proposing a compositional approach for automatically computing discrepancy for CPS. Consider a network $\mathcal{A}$ consisting of several interacting subsystems or modules $\mathcal{A}_1, \ldots, \mathcal{A}_N$. Our solution has several parts. First, we introduce a new type of input-to-state (IS) discrepancy function for the subsystems. An IS discrepancy for $\mathcal{A}_i$ gives a bound on the distance between two trajectories as a function of (a) their initial states and (b) the inputs they experience. For any positive parameter $\delta > 0$, using IS discrepancy of the modules we syntactically construct a reduced N-dimensional reduced system $M(\delta)$. We

show that the trajectory of $M(\delta)$ upper-bounds the distance of trajectories of $\mathcal{A}$ with initial states from a δ-ball. Moreover, by choosing appropriately small δ, the over-approximation computed by the above method can be made arbitrarily precise, modulo the precision of the numerical simulation.

Using the above results we develop an algorithm for bounded robust invariant verification for cyber-physical system models that iteratively refines initial set partitions (Algorithm 3.1). We show that the algorithm is sound and is guaranteed to terminate whenever the model is robustly safe or unsafe with respect to a given unsafe set. With a prototype implementation of this algorithm, we verify invariance properties for models with both linear and nonlinear dynamics [31, 33, 30]. Among these case studies are challenging pacemaker-heart models, where a pacemaker is regulating a group of cardiac cells [31]. Our implementation verifies such network with up to 8 cells and 32 continuous variables in a couple minutes.

1.4 Invariant Verification with Partial Order Reduction

The discrete behaviors of CPS are naturally modeled as *transitions*. For models with discrete transitions, an execution depends on the sequence of transitions it experiences. Hence, computing reach sets involves checking all possible transition sequences. In concurrent models, the transitions taken by different nodes (or processes) can be executed in different orders, which leads to a combinatorial explosion in the interleaving of possible transition sequences. Checking all these transition sequences explicitly is potentially very expensive. In Chapter 4, we use a discrete formalism of cyber-physical systems, namely labeled transition systems, to develop a theory of approximate partial order reduction for cyber-physical systems.

Partial order reduction was introduced in the context of model checking to reduce the number of executions that needed to be checked [34, 35]. It exploits the observation that some actions (or labels) can be executed in arbitrary order without affecting the result. Precisely, two actions a and b are *independent* if the resulting state would be the same regardless of the order in which a and b are executed. With the independence relation, any set of action sequences can be partitioned into a reduced set of equivalence classes, where equivalent action sequences can be made identical by swapping consecutive

independent actions. Many invariance properties have been shown to be indifferent to equivalent action sequences. Hence, checking invariance for a single execution suffices to infer it for all executions with equivalent action sequences.

In the context of CPS, components often interact with a shared environment in the presence of noise and disturbances. Independent action pairs in the conventional sense are rare, since those action pairs which lead to *nearly* but not *exactly* identical states are ignored. Our work addresses this problem by connecting the existing partial order reduction methods with metrics. First, we introduce the notion of ε-*independent actions* with a parameter ε chosen by the user: two actions a and b are ε-independent if the resulting states would be within ε distance regardless of the order in which they are executed. Similar to the conventional POR techniques, the approximate independence relation defines an equivalence relation of action sequences. Executions that follow equivalent action sequences reach not exactly but nearly the same final states. We present an algorithm to compute the upper bound on the distance between these executions. With this result, from a single execution, we can over-approximate the set of reachable states of executions following a class of equivalent action sequences. In addition, we show that by choosing small enough ε, the over-approximation can be made arbitrarily precise. Our approach potentially leads to an exponential reduction in complexity of the reach set over-approximation.

1.5 Differential Privacy for Distributed Control Systems

In many CPS, data about the individual participating agents can help achieve better system-level performance, but each individual's private data must be protected. Consider, for example, in a crowd-sourced traffic estimation and routing application as provided by Google maps and Waze [36, 9, 37, 38]. By explicitly exchanging information about their states, the agents could achieve better performance (routing delay), but by sharing exact information about their states they may give away too much information about their private data, such as their initial position and desired path. Examples include peak generation scheduling using consumption data obtained from smart electric

meters [39] and data aggregation for a sensor network [40, 41].

The *sensitive data* of agents in a cyber-physical system can be their initial states, final states or desirable trajectories. For a sensitive data set D, ξ_D denotes the corresponding trajectory of an individual agent. Although sharing the exact trajectory ξ_D throughout the network with other agents improves overall performance, it may leak the sensitive data of agents. One common approach for private state sharing is for each agent to add noise drawn from some carefully chosen distribution before sending. The effectiveness of such an approach can be measured by the concept of ε-*differential privacy* that developed in the study of databases [42, 43, 44, 45] and later extended to dynamical systems [46, 47]. Roughly, ε-differential privacy ensures that the probability distribution of the observation does not change substantially with the change in the sensitive data corresponding to one agent. To ensure a certain privacy level ε, the noise distribution must depend on the sensitivity of the trajectory ξ_D on the sensitive data D.

When noise is introduced to the system, the quality of communication deteriorates, influencing the performance of the system. In this thesis, we study the trade-off between ε-differential privacy and performance, in the context of discrete-time cyber-physical systems. Here, we measure the performance of the system with the mean square distance between the actual trajectory of an agent to its desired trajectory. For the discrete-time linear distributed control systems, we establish the trade-off between ε-differential privacy and the performance of the system. Specifically, the *cost of privacy*, namely the increase in the mean square error of the trajectories of an agent from its preference, up to T messages for a system with N agents is $O(\frac{T^3}{N\varepsilon^2})$ for stable systems and can also grow exponentially with T for unstable systems.

1.6 Overview of the Thesis

The techniques developed during the course of this PhD research have led to three main contributions to the analysis of cyber-physical systems.

(i) In Chapter 3, we propose an algorithm for verifying invariance of networked dynamical systems with communication delays. We introduce a notion of input-to-state (IS) discrepancy functions for components of

networks and a method for constructing a reduced model M of a networked dynamical system $\mathcal{A}$ using the IS discrepancy. Then, we show that the trajectory of the reduced model M upper-bounds the distance between trajectories of $\mathcal{A}$ with initial states close to each other. Moreover, the error can be made arbitrarily small modulo the precision of numerical simulation. Using the above results we develop an algorithm for bounded invariant verification of nonlinear networked dynamical systems that iteratively refines initial set partitions. We show that the algorithm is sound and complete for robust invariant verification.

(ii) In Chapter 4, we develop a partial order reduction method for infinite state labeled transition systems. We introduce the notion of ε-independent actions such that executing these actions in any order results in states that are close to each other. Then we define ε-equivalent action sequences that swap ε-independent action pairs. We present an algorithm to over-approximate reach sets of executions that take ε-equivalent action sequences. We are also able to show that the over-approximation can be computed up to arbitrary precision.

(iii) In Chapter 5, we propose a noise-adding mechanism that guarantees ε-differential privacy for discrete-time networked cyber-physical systems. Each agent masks its data with Laplace noise before sharing it with others for a better system-level control. The distributions of noise are specified by the sensitivity of the trajectory of agents to the sensitive data. We analyze the trade-off between ε-differential privacy and performance for linear cyber-physical systems. We show that the cost of ε-differential privacy is proportional to the cubic of time horizon and is inversely proportional to the number of participants and the square of the privacy level.

Reading this thesis. Chapters 3 to 5 are parallel but all rely on the basic definitions of Chapter 2. Each technical chapter has specific related work (Sections 3.2, 4.2 and 5.2) and model formalism (Sections 3.3, 4.3 and 5.3). Therefore the three technical chapters can be read independently.

Chapter 2

PRELIMINARIES

Many networked cyber-physical systems (NCPS) exhibit both continuous and discrete behaviors due to the complicated interactions between the computing units and the physical environment. In this thesis, we use several mathematical models with emphasis on different aspects of NCPS. In Chapter 3 we study *networked dynamical systems*, where the states evolve continuously over a period of time governed by *ordinary differential equations*. In Chapters 4 and 5, we introduce models of *transition systems*, where the states change instantaneously due to *discrete transitions*. Despite their distinctive syntaxes, these models share a common nature. Roughly, each of the models has a parameter domain, whose precise definition may vary across contexts, such as initial states, disturbances, action sequence, inputs, or control objectives. Given the value of its parameters, a model specifies a *trajectory*. In this chapter, we will introduce the notions used throughout the thesis.

2.1 Set, Metrics and Functions

For a natural number $n \in \mathbb{N}$, $[n]$ is the set of natural numbers $\{0, 1, \ldots, n-1\}$. For $p \in [1, \infty) \cup \{\infty\}$ and any $x \in \mathbb{R}^n$, $|x|_p$ is the standard ℓ^p norm of x. For a matrix $A \in \mathbb{R}^{m \times n}$, $|A|_p$ denotes the induced p norm of A, that is, $|A|_p \triangleq \sup_{|x|_p=1} |Ax|_p$. Without a subscript, $|x|$ and $|A|$ can be viewed as $|x|_p$ and $|A|_p$ for arbitrary choice of the constant $p \in [1, \infty]$.

For a vector $x \in \mathbb{R}^n$ and $r \geq 0$, $\mathcal{B}_r(x) \triangleq \{y \in \mathbb{R}^n \mid |x - y| \leq r\}$ is an r-neighborhood of x. For a set $S \subseteq \mathbb{R}^n$, the expansion of S is $\mathcal{B}_r(S) \triangleq \bigcup_{x \in S} \mathcal{B}_r(x)$. For any $i \in [n]$, x_i stands for the ith component of the vector x. For a compact set $S \subseteq \mathbb{R}^n$ and a $\delta > 0$, a δ-cover is a set of points $C \subseteq S$ where the δ-neighborhood of C covers S, that is $\mathcal{B}_\delta(C) = \cup_{x \in C} \mathcal{B}_\delta(x) \supseteq S$.

For any function f we denote the domain and the range of f by $dom(f)$ and

range(f). For any function f and a subset set of its domain $S \subseteq dom(f)$, we write the restriction of f to S as $f \restriction S : S \to range(f)$ such that $(f \restriction S)(x) = f(x)$ for any $x \in S$. For a function f whose range is a set of functions, we write $f \downarrow S$ such that for each $x \in dom(f)$, $f \downarrow S(x) = f(x) \restriction S$.

For $f, g : S \to \mathbb{R}^n$ with the same domain, we define $f + g : S \to \mathbb{R}^n$ such that $(f+g)(x) \triangleq f(x)+g(x)$. Similarly, we define $(f-g)(x) \triangleq f(x)+g(x)$ and $\max\{f, g\}(x) \triangleq \max\{f(x), g(x)\}$ similarly. For two functions f, g such that $dom(f) \supseteq range(g)$, the composition of f and g is $f \circ g : dom(g) \to range(f)$ such that $f \circ g(x) = f(g(x))$. We write $fg = f \circ g$ for brevity. For any $n \in \mathbb{N}$, for any function f maps from a domain to itself, that is $dom(f) \subseteq range(f)$, we define its nested form f^n as ff^{n-1} for $n \geq 1$ and f^0 being the identity mapping.

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is *smooth* if all its higher derivatives exist. It is said to be Lipschitz if there exists $L > 0$ such that $|f(x) - f(y)| \leq L|x - y|$ for all $x, y \in \mathbb{R}^n$. A function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ belongs to *class* $\mathcal{K}$, denoted $f \in \mathcal{K}$, if $f(0) = 0$ and f is strictly increasing. A class $\mathcal{K}$ function f belongs to *class* $\mathcal{K}_\infty$ if $f(x) \to \infty$ as $x \to \infty$. A function $f : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ belongs to *class* $\mathcal{KL}$ if (i) for any $y \in \mathbb{R}_{\geq 0}$, $f(x, y)$ belongs to class $\mathcal{K}$ with respect to argument x, and (ii) for any $x \in \mathbb{R}_{\geq 0}$, $f(x, y) \to 0$ as $y \to \infty$.

2.2 Time and Variables

We will use the notations from the hybrid input/uutput automaton (HIOA) framework for modeling networked cyber-physical systems [11, 12]. A *time domain* is a set $\mathbb{T} \in \{\mathbb{R}_{\geq 0}, \mathbb{N}\}$, where $\mathbb{R}_{\geq 0}$ is the *continuous time domain* and $\mathbb{N}$ is the *discrete time domain*. Fixed any time domain $\mathbb{T}$, an element of the time domain $t \in \mathbb{T}$ is a *time point*. A *time interval* $K \subseteq \mathbb{T}$ is a subset of the time domain if for any time points $t, t' \in K$, any time point lies between t and t' is also included in K. A time interval K is *left-closed* (*right-closed*) if it contains a minimum element (maximum element).

A *variable* is a name used to identify state components of a system and communication channels between components. Each variable v is associated with a type, *type*(v), which is the set of values v can take. A *valuation* for a set of variables V, maps each $v \in V$ with a value in *type*(v). For a set of variables V, *Val*(V) denotes the set of all possible valuations of V.

Valuations are denoted by $\mathbf{v}, \mathbf{x}, \mathbf{x}'$, etc. A variable v is *real-valued* if $\mathit{Val}(v)$ is an uncountable set of real numbers. A variable v is *finite-valued* if $\mathit{Val}(v)$ is a finite set. For a valuation $\mathbf{v}$ of V, the *restriction* of $\mathbf{v}$ to a set of variables $X \subseteq V$ is denoted by $\mathbf{v}.X \triangleq \mathbf{v} \lceil X$.

For a finite set of real-valued variables V with finite size $|V| = n$, a valuation $\mathbf{v}$ can be viewed as a vector in $\mathbb{R}^n$. Formally, an ordering of the set V is a bijection $O : V \to [n]$ that specify each variable to an index. Fixing an arbitrary ordering, a valuation $\mathbf{v}$ can be vectorized as a vector $x \in \mathbb{R}^n$, such that for any component $i \in [n]$, x_i is specified by the valuation of variable with index i, that is $x_i = \mathbf{v}(O^{-1}(i))$. We write $\mathit{vec}(\mathbf{v}) = x$ if the ordering is clear from the context. Using any vectorization, we translate the metric on $\mathbb{R}^n$ to the space $\mathit{Val}(V)$, such that for any pair of valuations $\mathbf{v}, \mathbf{v}'$, the metric $|\mathbf{v} - \mathbf{v}'| \triangleq |\mathit{vec}(\mathbf{v}) - \mathit{vec}(\mathbf{v}')|$. For $r \geq 0$, $\mathcal{B}_r(\mathbf{v}) \triangleq \{\mathbf{v}' \in \mathit{Val}(V) \mid |\mathbf{v} - \mathbf{v}'| \leq r\}$ is the closed neighborhood with radius r centered at $\mathbf{v}$. The notions of continuity, differentiability, and integration are lifted to functions defined over sets of valuations in the usual way. For a variable v, the *dynamic type* of v, $\mathit{dtype}(v)$, is a set of functions from left-closed intervals of its time domain $K \subseteq \mathbb{T}$ to $\mathit{type}(v)$. Roughly, the dynamic type of a variable specifies how its value can change overtime.

Example 2.1 (Variables). NCPS involves interaction between computing systems and physical environment, hence models of these systems may have variables with different static and dynamic types. In this thesis, we study both real-valued and finite-valued variables evolving in both continuous and discrete time domains. The real-valued continuous-time variables are useful for modeling the evolution of physical quantities such as temperature, voltage, velocity, etc. On the other hand, variables of software usually update in a discrete fashion. These variables can either be real-valued, such as the measurements of physical quantities, or be finite-valued, such as timers.

2.3 Trajectories

Consider a set of variables V where each variable $v \in V$ has an identical time domain $\mathit{time}(v) = \mathbb{T}$. A trajectory for a set of variables V describes the evolution of the values of the variables over a time interval. Precisely, a trajectory ξ of V is a function $\xi : K \to \mathit{Val}(V)$, where K is a left-closed

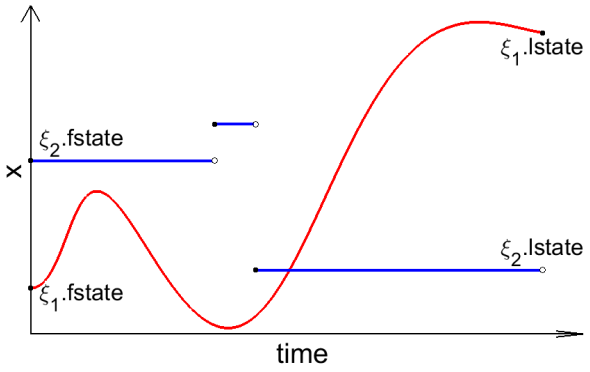


Figure 2.1: Example of trajectories of continuous-time variables. ξ_1 in red is a trajectory of real-valued continuous-time variables. ξ_2 in blue is a trajectory of finite-valued continuous-time variable.

interval of $\mathbb{T}$ with left endpoint equal to 0. For a subset of variables $S \in V$, $\xi \downarrow S : dom(\xi) \to Val(S)$ is a *projection* of the trajectory on variables S. We say that a trajectory ξ is finite if $dom(\xi)$ is upper bounded, closed if $dom(\xi)$ is (finite) right-closed. If ξ is finite, its limit time is the supremum of $dom(\xi)$, denoted as $\xi.\mathsf{ltime}$. We write the first state as $\xi.\mathsf{fstate} = \xi(0)$. Also if ξ is closed, we write the last state as $\xi.\mathsf{lstate} = \xi(\xi.\mathsf{ltime})$. For any $d \in \mathbb{T}$ and any trajectory ξ, a trajectory ξ' is the *d-delayed trajectory* of ξ if $\xi'(t) = \xi(0)$ for any $t \leq d$, and $\xi'(t) = \xi(t-d)$ for $t - d \in dom(\xi)$. For brevity, we write $\xi'(t) = \xi(t-d)$ for all $t \in dom(\xi')$.

Example 2.2 (Trajectories). As we discussed in Example 2.1, we consider four kinds of variables in this thesis. The variables associated with discrete-time domain $\mathbb{N}$ update in steps. Trajectories of these variables can be seen as sequences of valuations $\xi = \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T$. In contrast, continuos-time variables evolve continuously in time. We present example trajectories of continuous-time variables in Figure 2.1. ξ_1 is a trajectory of a real-valued continuous-time variable, which is piecewise continuous. In contrast, a trajectory of a finite-valued continuous-time variable, ξ_2, is piecewise constant.

2.4 Measure and Product Measure

In Chapter 5, we will discuss privacy of CPS based on a probabilistic model. The formulation and analysis of this problem involves measure theory. For a set S, a σ-algebra on S is a collection of subsets $\mathcal{F} \subseteq 2^S$, such that (i) $\mathcal{F}$ contains the empty set $\emptyset \in \mathcal{F}$, (ii) $\mathcal{F}$ is closed under complement and countable union. The pair $\langle S, \mathcal{F} \rangle$ is a *measurable space*. Any element $A \in \mathcal{F}$ in the σ-algebra is a *measurable set*. A function $\mu : \mathcal{F} \to \mathbb{R}_{\geq 0}$ is a *measure* on the measurable space $\langle S, \mathcal{F} \rangle$ if (i) the measure of empty set is 0, i.e. $\mu(\emptyset) = 0$, and (ii) μ is countably additive, such that for any collections $\{A_i\}_{i=0}^{\infty}$ of pairwise disjoint sets in $\mathcal{F}$, $\mu(\cup_{i=0}^{\infty} A_i) = \sum_{i=0}^{\infty} \mu(A_i)$. A measure μ is a *probability measure* if $\mu(S) = 1$. For a pair of measurable space $\langle S, \mathcal{F} \rangle$ and $\langle T, \mathcal{G} \rangle$, a function $f : S \to T$ is *measurable* if the pre-image of any measurable set $G \in \mathcal{G}$ is measurable. That is, for any $G \in \mathcal{G}$, the set $f^{-1}(G) = \{x \in S \mid f(x) \in G\}$ is in $\mathcal{F}$.

For a pair of measurable spaces $M_1 = \langle S_1, \mathcal{F}_1 \rangle$ and $M_2 = \langle S_2, \mathcal{F}_2 \rangle$, $M = M_1 \times M_2 = \langle S, \mathcal{F} \rangle$ is the *product measurable space* if (i) $S = S_1 \times S_2$ is the Cartesian product, and (ii) $\mathcal{F}$ is the complement and countable union closure of $\mathcal{F}_1 \times \mathcal{F}_2$. Suppose measurable spaces M_1 and M_2 are respectively equipped with measures μ_1 and μ_2. The *product measure* $\mu_1 \times \mu_2$ is a measure on the space $M_1 \times M_2$, where for any $A \in \mathcal{F}$, $\mu_1 \times \mu_2(A) = \int_{S_2} \mu_1(A(y)) d\mu_2(y)$ with $A(y) = \{x \in S_1 \mid (x, y) \in A\}$. We write μ^n as the product of n number of μ.

Chapter 3

VERIFYING INVARIANCE WITH INPUT-TO-STATE DISCREPANCY

Cyber-physical systems (CPS) are becoming commonplace in safety-critical applications. Many of these CPSs have geographically or computationally distributed components, and the communication between components may experience delays, which arise from transmission, processing, and buffering. In this chapter, we model these distributed cyber-physical systems (DCPS) as *networked dynamical systems*, where the states of the nodes evolve continuously as solutions of ordinary differential equations (ODEs). To aid in the design and development of such systems, we present algorithmic technique to verify invariance for networked dynamical systems in this chapter. A standard method for invariant verification involves computing the set of states visited by the trajectories (also called the *reach set*) of the system and checking whether the reach set satisfies the invariance. The problem of computing reach sets for a dynamical system is challenging because the initial set of these systems can be uncountable. Hence the number of trajectories is also uncountable, which makes explicit examination of all trajectories intractable.

Recently, several research groups developed simulation-based approaches to attain reach set over-approximation and invariant verification of dynamical systems [27, 28, 29, 30, 31, 48]. Roughly, a simulation-based algorithm computes the reach set of a dynamical system following several steps, as illustrated in Figure 3.1. First, the algorithm computes a finite cover of the initial set, each with a representative initial state. Then, for each cover S, the trajectory from the representative initial state θ is numerically simulated, namely ξ_θ. Based on the continuous dependence of trajectories to initial state, we know that all trajectories starting from the neighborhood of θ are close to ξ_θ. With sensitivity analysis of the trajectories to the initial state, we get quantitative bounds on the distance between trajectories from the cover S and ξ_θ. This enables us to compute a tube around ξ_θ, that contains all trajectories from the cover S. Repeating this process for all covers,

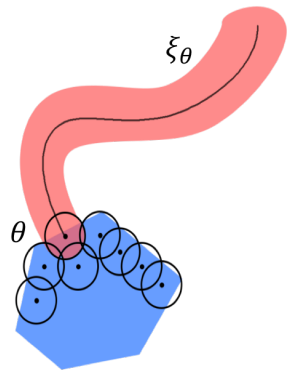


Figure 3.1: Simulation-based reach set over-approximation.

we can over-approximate the reach set of the dynamical system.

The precision or the conservativeness of this bound impacts the quality of the over-approximation. Ttherefore, the performance of verification with the above strategy. For example, the Lipschitz constant of the dynamic function f gives a bound on the distance between the neighboring trajectories that grows exponentially with time. Although checking Lipschitz continuity is generally undecidable, for certain classes of functions Lipschitz constants can be inferred from elementary functions [49]. Stronger notions like sensitivity [27], incremental Lyapunov functions [50], and contraction metrics [51] are used to obtain more practically useful bounds. In [29, 52], the authors generalized the above notions to a concept of *discrepancy functions*, which upper-bound the distance between trajectories using the distance between their initial states.

3.1 Invariant Verification with IS Discrepancy Functions

Central to simulation-based verification methods are the techniques for computing discrepancy functions (or for that matter, sensitivity, contraction metrics and incremental Lyapunov functions). However, there is no general method for finding discrepancy functions statically from the syntactic description of a dynamical system. One typically assumes a template poly-

nomial for the candidate function and then solves an optimization problem to find the coefficients. Recently in [48], the authors presented an on-the-fly approach for computing a version of locally valid discrepancy function, which only uses Lipschitz constants and Jacobian matrices.

However, both the static and the on-the-fly methods for computing discrepancy functions face several challenges in verifying networked dynamical systems. First, finding these discrepancy functions becomes increasingly difficult for larger networked dynamical systems in which many components interact. For example, the standard static approaches assume a template polynomial and solve for the coefficients. However the number of coefficients to solve can grow exponentially with the number of variables [32]. Same problem exists in the on-the-fly approach [48], which requires computing eigenvalues of Jacobian matrices. In practice, eigenvalues becomes harder to compute as the dimension of Jacobian increases. Second, none of the techniques we discussed above can be directly applied to networks with signal delays, while delays are pervasive due to message passing, buffering and computation. Last, in designing networked dynamical systems, one often encounters networks that have the same components but are interconnected in different topologies. The existing methods can merely compute discrepancy functions for these homogenous networks independently without utilizing their similarities.

In this chapter, we address these challenges by introducing a new compositional method for bounding distance between trajectories from initial states that are close to each other. We propose the notion of *input-to-state discrepancy function* (IS discrepancy function) for the components of the network. Consider a networked dynamical system $\mathcal{A}$ composed with N nodes $\{\mathcal{A}_i\}_{i\in[N]}$. Roughly, an IS discrepancy function (Definition 3.4) for a component $\mathcal{A}_i$ gives a bound on the distance between two trajectories of $\mathcal{A}_i$ as a function of (a) their initial states and (b) the inputs they experience.

Using IS discrepancy of the modules we syntactically construct a reduced N-dimensional dynamical system $M(\delta)$, where the parameter δ defines the initial state of M. If the interconnections in the original network $\mathcal{A}$ has delays then so do the interconnections in the reduced network. We show that $M(\delta)$ has a unique trajectory μ, which gives the discrepancy of the original network $\mathcal{A}$. Precisely, μ upper-bounds the distance between any pair of trajectories $\xi_{\mathbf{x}}$ and $\xi'_{\mathbf{x}}$ of network $\mathcal{A}$, where the initial states $\mathbf{x}$ and $\mathbf{x}'$ are within δ distance (Theorem 3.7). Thus, by simulating $\mathcal{A}$ and (the smaller) $M(\delta)$

we can compute the bounded-time reach set over-approximations of $\mathcal{A}$. We also show that by choosing appropriately small δ's the over-approximations computed by the above method can be made arbitrarily precise; modulo the precision of the numerical simulations (Theorem 3.10).

Using the above results we develop an algorithm for bounded safety verification of nonlinear dynamical systems that iteratively refines initial set partitions (Algorithm 3.1). We show that the algorithm is sound and is guaranteed to terminate whenever the model is robustly safe or unsafe with respect to a given unsafe set. Our experiments with a prototype implementation of the algorithm show that the approach achieves best performance when verifying networks of stable modules with light inter-modular couplings.

The rest of this chapter is arranged as follows. First in Section 3.2 we discuss related works. In Section 3.3, we formally introduce the model of networked dynamical systems. We present the definition of the IS discrepancy function in Section 3.4 and the construction of a reduced model with IS discrepancy functions in Section 3.5. We develop an algorithm to compute reach set over-approximation with the reduced model in Section 3.6. The algorithm is then applied to verify invariance for linear and nonlinear networked dynamical system models in Section 3.7.

3.2 Related Works

Invariant verification of cyber-physical systems has been actively studied in the past two decades [53, 54, 11]. A most commonly used approach for verifying invariance is through computing reach sets. Since the problem of computing precise reach sets of dynamical system is in general undecidable [55], many researchers shift to techniques for over-approximating reach sets. Reach sets of a nonlinear system can be over-approximated using abstractions, which are interpretations of the original system with more behaviors and simpler dynamics [56, 57, 58]. Examples of these abstractions include discrete transition systems [57] and multi-affine systems [56]. If an abstraction satisfies the invariant, so does the original model. However, one cannot conclude the inverse if the abstraction violates the invariant. To resolve that, counter-example-guided abstraction refinement (CEGAR) procedure [58] is adopted to construct finer abstractions. Although this technique has been successful

in many invariant verification case studies, it suffers from the curse of dimensionality as constructing abstractions often involves state space partitioning.

Simulation-based verification techniques over-approximate the reach set of dynamical and hybrid systems using numerically simulated behaviors [59, 27, 29]. Finitely many simulations alone cannot prove the invariance property. With sensitivity analysis, a single trajectory can embody trajectories with neighboring initial states [27, 60]. This idea is then generalized to the notion of discrepancy between trajectories, which enables a sound and relatively complete invariant verification algorithm [29, 52, 48].

In [48], the authors developed an on-the-fly approach for computing a local version of discrepancy that uses only the Lipschitz constants and statically computed Jacobian matrices. The approach requires estimating upper bounds on the eigenvalue of the symbolically computed Jacobian of the whole system. However, in this chapter, we focus on a compositional approach for computing discrepancy function for networked systems. Our approach can be combined with the on-the-fly computation for computing a local version of IS discrepancy that uses only the Lipschitz constants and the Jacobian matrices of the modules [61].

There has been recent work on verification of delayed differential equations (see, for example, [62] and the references therein). The algorithm in [62] iteratively computes validated simulations of delayed differential equations as sequences of Taylor over-approximations of time intervals, and verifies safety property using SMT solvers. Part of its technique can be used in our approach as a mean for computing validated numerical simulations for delayed differential equations. Our approach, in addition, involves systematically generating numerical simulations to refine reach sets and dynamically reasoning about sensitivity of interconnecting networks.

3.3 Networked Dynamical System Models

Networked dynamical systems are built-up by composing smaller component models. In this section, we define component models and the composition operation. The composition operation we use can model the communication delay between components.

3.3.1 Dynamical System Modules

The *dynamical system module* is the building block for networked dynamical systems. A dynamical system module is specified by a collection of ordinary differential equations (ODEs), possibly with inputs, and a set of initial states. For reducing notational overhead, we identify output variables with state variables in this chapter but our results can be extended to systems where outputs are distinct in a straightforward manner.

Definition 3.1. *A* dynamical module $\mathcal{A}$ *is a tuple* $\langle X, U, \Theta, f \rangle$ *where*

(i) X *is a set of variables called the* state variables*; valuations of* X *are called* states*;*

(ii) U *is a set of variables called the* input variables *that are distinct from the state variables;*

(iii) $\Theta \subseteq \mathit{Val}(X)$ *is a compact set of* initial states*;*

(iv) $f : \mathbb{R}_{\geq 0} \times \mathit{Val}(X) \times \mathit{Val}(U) \to \mathit{Val}(X)$ *is called the* dynamic mapping*. In addition,* f *is continuous in the first argument and Lipschitz continuous with respect to the other two arguments.*

As we discussed in Chapter 2, valuations of the state variables $\mathbf{x} \in \mathit{Val}(X)$ and the input variables $\mathbf{u} \in \mathit{Val}(U)$ can be seen respectively as vectors in $\mathbb{R}^{|X|}$ and $\mathbb{R}^{|U|}$. The *space of input signal* ($\mathcal{U}$) is the set of all continuous trajectories of the input variables U. Fixing any input signal $\eta \in \mathcal{U}$ and any initial state $\theta \in \Theta$, a trajectory of $\mathcal{A}$ is uniquely specified of $\mathcal{A}$ is $\xi_{\theta,\eta} : \mathbb{R}_{\geq 0} \to \mathit{Val}(X)$ such that (i) $\xi(0) = \theta$, and (ii) for any $t \in \mathbb{R}_{\geq 0}$, the derivative of ξ satisfies the differential equation

$$\dot{\xi}(t) = f(t, \xi(t), \eta(t)). \tag{3.1}$$

As in Equation (3.1), we will suppress the subscripts of ξ when the dependence on the initial state and the input trajectory are clear from context. Safety verification problems commonly involve computing the set of trajectories up to some bounded time $T > 0$. The (global) Lipschitz assumption of the dynamic mapping f and the piecewise continuity of η guarantee that the differential Equation (3.1) admits a unique global solution for interval $[0, T]$ with any time bound $T > 0$ and any initial state $\theta \in \mathit{Val}(X)$. In case the

trajectories of the system are confined to an invariant set $S \subseteq Val(X)$, the results in this chapter would hold for a locally Lipschitz continuous f with L as the Lipschitz constant over S.

A module $\mathcal{A}$ without inputs ($U = \emptyset$) is said to be *closed*; otherwise, $\mathcal{A}$ is *open*. The set of all trajectories of $\mathcal{A}$ with respect to a set of initial states $\Theta' \subseteq \Theta$ and a set of input signals $\mathcal{U}' \subseteq \mathcal{U}$ is denoted by $Traj(\mathcal{A}, \Theta', \mathcal{U}')$. We will drop the argument $\mathcal{U}'$ for closed modules. The components of modules $\mathcal{A}$ and $\mathcal{A}_i$ are denoted by $X_{\mathcal{A}}, U_{\mathcal{A}}, \Theta_{\mathcal{A}}, f_{\mathcal{A}}$ and X_i, U_i, Θ_i, f_i, respectively.

Example 3.1 (FHN). The FitzHugh-Nagumo (FHN) model [63] is a generic model for excitable media and can be applied to cardiac cells. The model has two state variables $X = \{x, v\}$, where x models the membrane voltage of a cardiac cell and v is an internal variable. Initially, the variables x, v take values in the ranges $[1.4, 1.6]$ and $[0, 0.2]$, respectively. The input variables for an FHN model are $U = \{u_1, u_2, u_3\}$, where u_1 and u_2 model the voltages of two neighboring cells and u_3 models the input pulse form a pacemaker. The dynamic of the model is:

$$\begin{aligned} \dot{x} &= -x^3 + 1.1x^2 - 0.12x - v + 0.01u_1 + 0.01u_2 + u_3 \\ \dot{v} &= 0.005x - 0.01v. \end{aligned} \tag{3.2}$$

Next, for a closed module $\mathcal{A}$ we define reachable states and safety. A state $\mathbf{x} \in Val(X)$ is *T-reachable* if there exists a trajectory $\xi \in Traj(\mathcal{A}, \Theta)$ and a time $t \leq T$ such that the trajectory $\xi(t) = \mathbf{x}$. The set of T-reachable states is denoted by $\mathsf{Reach}_{\mathcal{A}}(\Theta, [0, T]) \triangleq \{\xi(t) \mid t \leq T, \xi \in Traj(\mathcal{A}, \Theta)\}$.

Definition 3.2. *For $\epsilon \geq 0$ and time $T \geq 0$, and an open unsafe set $Unsafe \subseteq Val(X)$, $\mathcal{A}$ is* ϵ-robustly safe *up to T if $\mathcal{B}_\epsilon(\mathsf{Reach}_{\mathcal{A}}(\Theta, [0, T])) \cap Unsafe = \emptyset$. If there exists some $\epsilon > 0$ for which this condition holds, then $\mathcal{A}$ is* robustly safe up to T with respect to *Unsafe*.

3.3.2 Networked Dynamical Systems with Delays

Formally, the composition operation takes a collection of modules and defines a new dynamical system by plugging-in or identifying the input variables of one subsystem with state variables of another. The resulting system may still have input variables that are not identified with any of the state variables.

A collection of dynamical modules $\mathcal{A} = \{\mathcal{A}_i\}_{i\in[N]}$ are *compatible* if they do not share any of the state variables. That is, for any $i, j \in [N]$, $X_i \cap X_j = \emptyset$. This condition merely prevents any unforeseen identification of states and inputs. The collection is *closed* if $\cup_{i\in[N]} U_i \subseteq \cup_{i\in[N]} X_i$; that is, as a whole, the network has no free input variable. In this chapter, we will develop compositional techniques for analyzing closed networks.

In general, when the output signals from one module are plugged into the input of another module, the latter may receive these inputs with some delay. Delays arise from transmission, processing, and buffering. In this chapter, we assume that all the signals from one module to another are identically delayed. The signals from the first module to a third module may be delayed by a different amount from the delay experience by the second module. This is a reasonable assumption when the delays are dominated by module-level effects as opposed to the individual variable-related effects.

For a collection of N modules $\{\mathcal{A}_i\}_{i\in[N]}$, a *delay matrix* $d \in {\mathbb{R}_{\geq 0}}^{N\times N}$ gives for each ordered pair (i, j) a non-negative time-delay constant $d(i, j)$. All the input signal from module $\mathcal{A}_i$ to module $\mathcal{A}_j$ experience the identical delay of $d(i, j)$. Now that we have defined dynamical modules and the delay matrix for interconnecting them, we proceed to defining dynamic networks with delays.

Definition 3.3. *For a compatible and closed collection of dynamical modules* $\{\mathcal{A}_i\}_{i\in[N]}$*, and a delay matrix* $d \in {\mathbb{R}_{\geq 0}}^{N\times N}$*, a* networked dynamical system*, denoted by* $\mathcal{A} = \|_d\{\mathcal{A}_i\}_{i\in[N]}$*, is the tuple* $\langle X, \Theta, \mathcal{T}\rangle$*, where*

(i) $X = \cup_{i\in[N]} X_i$*,*

(ii) $\Theta = \{\theta \in \mathit{Val}(X) \mid \theta \lceil X_i \in \Theta_i, \text{ for each } i\}$*, and*

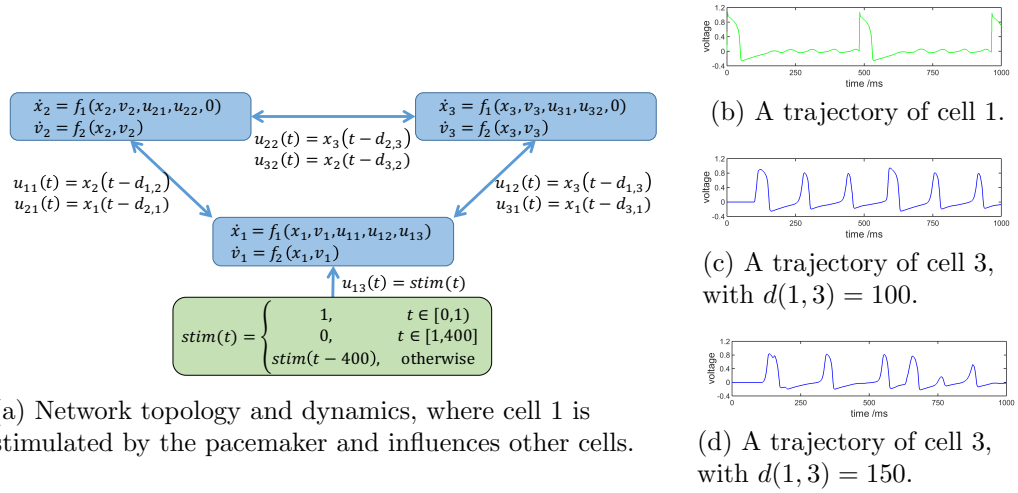
(iii) $\mathcal{T}$ *is a set of trajectories of* $\mathcal{A}$ *such that* $\xi \in \mathcal{T}$ *if and only if for each* $i \in [N]$*,* $t \in \mathbb{R}_{\geq 0}$*,*

$$\dot{\xi}_i(t) = f_i(t, \xi_i(t), \eta_i(t)), \tag{3.3}$$

where $\xi_i = \xi \downarrow X_i$ *and for each input variable* $u \in U_i \cap X_j$ *from* j *to* i*,* $\eta_i(t) \lceil u = \xi_j(t - d(i, j)) \lceil u$*.*

It can shown that any initial state $\theta \in \Theta$ uniquely specifies a trajectory of $\mathcal{A}$. We refer readers to [64] for a general discussion on the existence and uniqueness of solutions to delayed differential equations. Here we sketch the

argument. Recall from Section 2.3 that, for any $i, j \in [N]$ and any time $t < d(i,j)$, we defined the time delay trajectory as $\xi_j(t - d(i,j)) = \xi_j(0)$. For a networked dynamical system $\mathcal{A}$ defined above, let $d_M = \max_{i,j} d(i,j)$ be the maximum delay and $\mathcal{C} = C([-d_M, 0], \mathbb{R}^{|X|})$ be the space of continuous functions from $[-d_M, 0]$ to $\mathbb{R}^{|X|}$. Let $H_{\xi,d_M,t} \in \mathcal{C}$ be the history of trajectory ξ of length d_M such that $H_{\xi,d_M,t}(s) = \xi(t+s)$, for all $s \in [-d_M, 0]$. The trajectories of the network defined in Equation (3.3) are solutions of the delayed differential equation $\dot{\xi}(t) = f(t, H_{\xi,d_M,t})$ with $f : \mathbb{R} \times \mathcal{C} \to \mathbb{R}^{|X|}$ such that f is an aggregate of all the N Equations (3.3). Since each f_i is continuous in t and Lipschitz in the other arguments, we can check that f is continuous in t and Lipschitz in $H_{\xi,d_M,t}$.[1] From Theorems 2.1 and 2.2 in [64], for any initial state $\theta \in \Theta$, a networked dynamical system $\mathcal{A}$ has a unique trajectory ξ. With the set of trajectories $Traj(\mathcal{A}, \Theta)$ as defined above, the set of bounded time reachable states $\mathsf{Reach}_{\mathcal{A}}(\Theta, [0,T])$ of the closed networked dynamical system $\mathcal{A}$ is defined analogously to that of closed dynamical modules.



(a) Network topology and dynamics, where cell 1 is stimulated by the pacemaker and influences other cells.

(b) A trajectory of cell 1.

(c) A trajectory of cell 3, with $d(1,3) = 100$.

(d) A trajectory of cell 3, with $d(1,3) = 150$.

Figure 3.2: A networked dynamical system with three cells and a pacemaker and its sample trajectories.

Example 3.2 (Pacemaker-Heart Model). We consider a network of cardiac cells $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ stimulated by a pacemaker, as illustrated in Figure 3.2(a). The cardiac cells follow the FHN model presented in Example 3.1. The network is defined by associating input of each cell $\mathcal{A}_i$ to the state of another $\mathcal{A}_j$ with a non-negative delay $d_{i,j}$. The pacemaker is modeled as a rectangular

[1] The norm on the space $\mathcal{C}$ is the L^∞ norm defined as $|H| = \sup_{t \in [-d_M, 0]} |H(t)|$.

wave generator with period 450 and duty cycle 2. We simulate the network by assigning the delays $d(1,2) = d(2,1) = 70$, $d(2,3) = d(3,2) = 40$ and $d(3,1) = 100$ with two different delay values of $d(1,3)$. A trajectory of cell 1 with $d(1,3) = 100$ is plotted in Figure 3.2(b). Two trajectories of cell 3 with $d(1,3) = 100$ and $d(1,3) = 150$ are plotted in (c) and (d), respectively. Not only the trajectory of cell 3 is shifted in time due to different delay value, but the shape of the trajectories also changes due to the combined inputs from cells 1 and 2.

3.4 Input-to-State Discrepancy

In this section, we introduce the notion of input-to-state (IS) discrepancy functions for dynamical system modules and present three approaches for finding them. In Section 3.5, we will use IS discrepancy to develop the approaches for computing over-approximations of reachable states.

3.4.1 IS Discrepancy and Witnesses

Roughly, input-to-state discrepancy (IS discrepancy) of a module $\mathcal{A}$ bounds the distance between two trajectories of $\mathcal{A}$ in terms of different initial states and different inputs.

Definition 3.4. *For a dynamical module $\mathcal{A} = \langle X, U, \Theta, f \rangle$, a continuous function $V : Val(X)^2 \to \mathbb{R}_{\geq 0}$ is an* input-to-state discrepancy function *if*

(i) *$\exists$ class-$\mathcal{K}$ functions $\underline{\alpha}, \overline{\alpha}$ such that for any $\mathbf{x}, \mathbf{x}' \in Val(X)$, $\underline{\alpha}(|\mathbf{x}-\mathbf{x}'|) \leq V(\mathbf{x}, \mathbf{x}') \leq \overline{\alpha}(|\mathbf{x}-\mathbf{x}'|)$, and*

(ii) *$\exists \beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ of class-$\mathcal{K}$ in the first argument and a class-$\mathcal{K}$ function γ such that for any pair of initial states $\theta, \theta' \in \Theta$, and pair of input trajectories $\eta, \eta' \in Traj(U)$, and $t \in \mathbb{R}_{\geq 0}$,*

$$V(\xi(t), \xi'(t)) \leq \beta(|\theta - \theta'|, t) + \int_0^t \gamma(|\eta(s) - \eta'(s)|)ds, \tag{3.4}$$

where $\xi = Traj(\mathcal{A}, \theta, \eta)$ and $\xi' = Traj(\mathcal{A}, \theta', \eta')$.

$(\underline{\alpha}, \overline{\alpha}, \beta, \gamma)$ are called the witnesses *of the IS discrepancy function.*

The norm $|\cdot|$ is the standard p-norm with $p \in [1, \infty]$ chosen by the user. In the rest of the chapter, we make a technical assumption that $\underline{\alpha}^{-1}$ and γ are Lipschitz, and $\beta(\cdot, \cdot)$ has a Lipschitz continuous derivative in the second argument. These assumptions enable us to construct a reduced model with well-defined trajectories in Section 3.5.1. The discrepancy function (and its witnesses) bounds the maximum distance between two trajectories in terms of the ℓ^2 distance between their input signals and their initial states. This type of discrepancy function is motivated by the notion of incremental integral input-to-state stability of dynamical modules [65], except that we do not require that for any $x \in \mathbb{R}_{\geq 0}$ that $\beta(x, t)$ in Equation (3.4) converges to 0 as $t \to \infty$. We assume that the witnesses $(\underline{\alpha}, \overline{\alpha}, \beta, \gamma)$ are Lipschitz continuous.

3.4.2 Finding IS Discrepancy

In what follows, we describe three well-known methods for obtaining input-to-state stability proofs and here we note that they can be used to find IS discrepancy functions.

Lipschitz Dynamics. For any dynamical module $\mathcal{A}$ with Lipschitz continuous dynamic mapping f and for any bounded time, we can find an IS discrepancy function of $\mathcal{A}$ for that time bound. This version of IS discrepancy will be sufficient for bounded safety proofs. We note that the problem of computing Lipschitz constant for a function is generally undecidable. However, for many elementary functions such as trigonometric functions and polynomial functions, the Lipschitz constant can be computed for any compact set [49]. There are also some heuristics to estimate Lipschitz constant for general functions [66].

Proposition 3.1. *Suppose the dynamic mapping f of module $\mathcal{A}$ is Lipschitz in both arguments with Lipschitz constant L. For any time bound $T > 0$, $V(\mathbf{x}, \mathbf{x}') \triangleq |\mathbf{x} - \mathbf{x}'|$ is a discrepancy function with witnesses $(\underline{\alpha}, \overline{\alpha}, \beta, \gamma)$ where $\underline{\alpha}(|\mathbf{x} - \mathbf{x}'|) = \overline{\alpha}(|\mathbf{x} - \mathbf{x}'|) = |\mathbf{x} - \mathbf{x}'|$, $\beta(|\theta - \theta'|, t) = e^{Lt}|\theta - \theta'|$ and $\gamma(|\mathbf{u} - \mathbf{u}'|) = Le^{LT}|\mathbf{u} - \mathbf{u}'|$.*

Stable Linear Dynamics. Suppose $\mathcal{A}$ has dynamic mapping $f(\mathbf{x}, \mathbf{u}) = C\mathbf{x} + D\mathbf{u}$, where C is a $n \times n$ matrix and D is a $n \times m$ matrix. If C is

asymptotically stable, then its input-free trajectories converge exponentially and we can get an IS dependency function with exponentially convergent witness β.

Proposition 3.2. *For linear module $\mathcal{A}$ with dynamic mapping $f(\mathbf{x}, \mathbf{u}) = C\mathbf{x} + D\mathbf{u}$ with asymptotically stable matrix C, there exists $\lambda > 0$, such that $V(\mathbf{x}, \mathbf{x}') = |\mathbf{x} - \mathbf{x}'|$ is a discrepancy function with witnesses $(\underline{\alpha}, \overline{\alpha}, \beta, \gamma)$ where $\underline{\alpha}(|\mathbf{x}-\mathbf{x}'|) = \overline{\alpha}(|\mathbf{x}-\mathbf{x}'|) = |\mathbf{x}-\mathbf{x}'|$, $\beta(|\theta-\theta'|, t) = e^{-\lambda t}|\theta-\theta'|$ and $\gamma(|\mathbf{u}-\mathbf{u}'|) = |D||\mathbf{u} - \mathbf{u}'|$.*

The positive constant λ can be computed by solving Lyapunov equation [67].

Incremental Integral ISS Dynamics. The notion of incremental integral input-to-state stability (incremental integral ISS) of dynamical modules [65] is a generalization of the standard notion of input-to-state stability [68, 69, 50]. A Lyapunov-like theorem of proving incremental integral ISS as well as a converse Lyapunov theorem are presented in [65]. Given a proof of an incremental integral ISS property of a module, we automatically get its IS discrepancy function (with witnesses).

Definition 3.5. *A dynamical module $\mathcal{A}$ is called incremental-integral-input-to-state stable (δiISS) if there exists a class-$\mathcal{K}_\infty$ function α, a class-$\mathcal{KL}$ function β and a class-$\mathcal{K}$ function γ such that, for any initial states $\theta, \theta' \in \Theta_\mathcal{A}$, for any input signal $\eta, \eta' \in U_\mathcal{A}$ and any $t > 0$,*

$$\alpha(|\xi(t) - \xi'(t)|) \leq \beta(|\theta - \theta'|, t) + \int_0^t \gamma(|\eta(s) - \eta'(s)|)ds, \tag{3.5}$$

where $\xi = Traj(\mathcal{A}, \theta, \eta)$ and $\xi' = Traj(\mathcal{A}, \theta', \eta')$.

Proposition 3.3. *Given an incremental integral ISS module $\mathcal{A}$ with (α, β, γ) as in Definition 3.5, then $V(\mathbf{x}, \mathbf{x}') = \alpha(|\mathbf{x} - \mathbf{x}'|)$ is a discrepancy function with witnesses $(\underline{\alpha}, \overline{\alpha}, \beta, \gamma)$ where $\underline{\alpha}(|\mathbf{x} - \mathbf{x}'|) = \overline{\alpha}(|\mathbf{x} - \mathbf{x}'|) = \alpha(|\mathbf{x} - \mathbf{x}'|)$, and β, γ given above.*

Proposition 3.1 establishes an IS discrepancy function only using the Lipschitz constant of the dynamic mapping f. Although, computing Lipschitz constants of arbitrary functions is undecidable in general, for certain classes

of polynomial functions it can be estimated [49]. However, even if the dynamics of the module is stable, IS discrepancy functions obtained by this method have witnesses β and γ that grow exponentially with time. Incremental-integral-input-to-state stability (Definition 3.5) can be proved by constructing an incremental Lyapunov function [65]. For some dynamical models with physical implications, the incrememtal Lyapunov function may capture the energy of the system. However, constructing the incremental Lyapunov function is in general a non-trivial problem.

For modules with linear dynamics, IS discrepancy can be computed automatically using Proposition 3.2. For nonlinear modules, we refer the readers to Algorithm 2 of [61], which is an algorithm that computes a local version of discrepancy function and witnesses that satisfy all these assumptions.

3.5 Small Approximations from IS Discrepancy

In this section we construct a scalar (one-dimensional) approximation for each $|X_i|$-dimensional dynamical module $\mathcal{A}_i = \langle X_i, U_i, \Theta_i, f_i \rangle$ (Definition 3.6) in a network, using input-to-state discrepancy functions. For the sake of a cleaner presentation, we develop the results for a network consisting of two $|X_1|$ and $|X_2|$-dimensional modules $\mathcal{A}_1$ and $\mathcal{A}_2$ with a two-dimensional approximation. The general results follow by straightforward extension and are stated in Section 3.5.4.

3.5.1 IS Approximation of $\|_d\{\mathcal{A}_1, \mathcal{A}_2\}$

Consider a closed networked dynamical system $\mathcal{A} = \|_d\{\mathcal{A}_1, \mathcal{A}_2\}$ composed of two modules $\mathcal{A}_1 = \langle X_1, U_1, \Theta_1, f_1 \rangle$ and $\mathcal{A}_2 = \langle X_2, U_2, \Theta_2, f_2 \rangle$ and a delay matrix $d \in {\mathbb{R}_{\geq 0}}^{2\times 2}$. The input signals U_2 of $\mathcal{A}_2$ are obtained from X_1 delayed by $d(2,1)$ and the input signals U_1 of $\mathcal{A}_1$ are obtained from X_2 delayed by $d(1,2)$. Let V_i be an IS discrepancy function for $\mathcal{A}_i, i \in \{1,2\}$ with witness $(\underline{\alpha}_i, \overline{\alpha}_i, \beta_i, \gamma_i)$. For any pair of initial states θ, θ' in $\Theta_{\mathcal{A}}$, let $\xi = Traj(\mathcal{A}, \theta)$ and $\xi' = Traj(\mathcal{A}, \theta')$ be the unique trajectories of the network $\mathcal{A}$ starting from θ and θ', respectively. We define $\theta_i = \theta \lceil X_i$, $\theta_i' = \theta' \lceil X_i$, $\xi_i = \xi \downarrow X_i$ and $\xi_i' = \xi' \downarrow X_i$. From Definition 3.3, the restriction of ξ_i and ξ_i' to X_i are trajectories of $\mathcal{A}_i$ from θ_i and θ_i'. From Definition 3.4, for every $t \in [0, T]$ the

following holds:

$$V_1(\xi_1(t), \xi_1'(t)) \leq \beta_1(|\theta_1 - \theta_1'|, t) + \int_0^t \gamma_1(|\xi_2(s - d(1,2)) - \xi_2'(s - d(1,2))|)ds,$$
$$V_2(\xi_2(t), \xi_2'(t)) \leq \beta_2(|\theta_2 - \theta_2'|, t) + \int_0^t \gamma_2(|\xi_1(s - d(2,1)) - \xi_1'(s - d(2,1))|)ds. \tag{3.6}$$

Recall that when a trajectory ξ is evaluated at $t < 0$, we use the valuation $\xi(t) = \xi(0)$. Next, we introduce the key notion of a family of IS approximations for $\mathcal{A}$. Each approximation is parameterized by non-negative reals $\delta_1, \delta_2 \geq 0$ and is a closed networked dynamical system M with two main variables m_1 and m_2. As we shall see in Theorem 3.7, at any time t, m_1 gives an upper bound on the distance between two state trajectories of $\mathcal{A}_1$ that start from neighboring states at most δ_1 apart. Similarly, m_2 gives an upper bound on neighboring trajectories of $\mathcal{A}_2$. Of course, the distance between two neighboring state trajectories of $\mathcal{A}_1$ depends on (a) their initial states and (b) on the inputs they experience. These inputs in turn are delayed versions of the state trajectories of $\mathcal{A}_2$. So, the dynamics of m_1 (and m_2) takes into account the impact of both of these factors using the witnesses of the IS discrepancy functions. Since β_1 and β_2 witnesses bound the impact of initial states on the discrepancy as a function of time, the dynamics of m_1 (and m_2) are time varying.

Definition 3.6. *For a pair of non-negative constants (δ_1, δ_2), the (δ_1, δ_2)-*IS approximation *of a networked dynamical system $\mathcal{A} = \|_d\{\mathcal{A}_1, \mathcal{A}_2\}$ is a closed networked dynamical system $M = \langle X_M, \Theta_M, \mathcal{T}_M \rangle$ where*

(i) $X_M = \{m_1, m_2\}$,

(ii) $\Theta_M = \{\theta\}$ *where* $\theta \lceil m_i = \beta_i(\delta_i, 0)$, *for* $i \in \{1, 2\}$, *and*

(iii) for any trajectory μ of X_M, $\mu \in \mathcal{T}_M$ if and only if

$$\begin{aligned} \dot{\mu}_1(t) &= \dot{\beta}_1(\delta_1, t) + \gamma_1 \underline{\alpha}_2^{-1}(\mu_2(t - d(1,2))), \\ \dot{\mu}_2(t) &= \dot{\beta}_2(\delta_2, t) + \gamma_2 \underline{\alpha}_1^{-1}(\mu_1(t - d(2,1))). \end{aligned} \tag{3.7}$$

Here $\mu_i = \mu \downarrow m_i$ and $\dot{\beta}_i(\delta_i, t)$ is a short hand for $\frac{\partial}{\partial t}\beta_i(\delta_i, t)$.

Recall from Section 2.1 that $\gamma_1\underline{\alpha}_2^{-1}$ and $\gamma_2\underline{\alpha}_1^{-1}$ are the function compositions $\gamma_1 \circ \underline{\alpha}_2^{-1}$ and $\gamma_2 \circ \underline{\alpha}_1^{-1}$, respectively. The dynamics of the closed reduced network M is defined syntactically in terms of the IS discrepancy functions of the dynamic modules $\mathcal{A}_1$ and $\mathcal{A}_2$ and the delay matrix. The witness functions $\underline{\alpha}_1^{-1}, \gamma_1$, etc., are Lipschitz continuous and Lipschitz functions are closed under composition. Therefore, the delay differential Equation (3.7) has a unique solution. Since M has a single initial state, it is completely deterministic. Note that both the initial state and the dynamics of M depend on the choice of the parameters δ_1 and δ_2. In Theorem 3.7 we relate m_1 and m_2 with the divergence between trajectories of $\mathcal{A}_1$ and $\mathcal{A}_2$. Specifically, if μ is the trajectory of a (δ_1, δ_2)-IS approximation then $\mu_i = \mu \downarrow m_i(t)$ gives an upper bound on the distance between the trajectories of $\mathcal{A}_i$ starting from initial states that are at most δ_i apart.

3.5.2 Over-Approximation with IS Discrepancy

For any pair of non-negative reals $\delta = (\delta_1, \delta_2)$ and any state $\mathbf{x} \in \mathit{Val}(X_\mathcal{A})$, we define

$$\mathcal{B}_\delta(\mathbf{x}) = \mathcal{B}_{\delta_1}(\mathbf{x} \lceil X_1) \times \mathcal{B}_{\delta_2}(\mathbf{x} \lceil X_2)$$

as the product of the δ_i-balls around $\mathbf{x} \lceil X_i$. Given a pair of discrepancy functions $V = (V_1, V_2)$ for $\mathcal{A}_1$ and $\mathcal{A}_2$, a state $\mathbf{m} \in \mathit{Val}(X_M)$ of M naturally defines a sublevel set $L_V(\mathbf{m}) \subseteq \mathit{Val}(X_A) \times \mathit{Val}(X_A)$:

$$L_V(\mathbf{m}) = \{(\mathbf{x}, \mathbf{x}') \mid \forall i \in \{1, 2\},\ V_i(\mathbf{x} \lceil X_i, \mathbf{x}' \lceil X_i) \leq \mathbf{m} \lceil m_i\}.$$

This set is the intersection of the $(\mathbf{m} \lceil m_i)$-*sublevel sets* of V_i. For a state $\mathbf{x} \in \mathit{Val}(X_\mathcal{A})$ of $\mathcal{A}$ and a state $\mathbf{m} \in \mathit{Val}(X_M)$ we define

$$\mathcal{B}^V_\mathbf{m}(\mathbf{x}) = \{\mathbf{x}' \in \mathit{Val}(X_\mathcal{A}) \mid (\mathbf{x}, \mathbf{x}') \in L_V(\mathbf{m})\}$$

as the subset of states of $\mathcal{A}$ for which $(\mathbf{x}, \mathbf{x}')$ lies in the sublevel set defined by $\mathbf{m}$.

We will now prove a sequence of results that ultimately established in Lemma 3.6 that the trajectory of M gives an upper bound on the discrepancy

of $\mathcal{A}$. That is, at any time $t \geq 0$,

$$\begin{aligned}\beta_1(|\theta_1 - \theta_1'|, t) + \int_0^t \gamma_1(|\xi_2(s - d(1,2)) - \xi_2'(s - d(1,2))|)ds \leq \mu_1(t) \\ \beta_2(|\theta_2 - \theta_2'|, t) + \int_0^t \gamma_2(|\xi_1(s - d(2,1)) - \xi_1'(s - d(2,1))|)ds \leq \mu_2(t).\end{aligned} \tag{3.8}$$

From the construction of M, we observe that at time $t = 0$, the above inequalities hold. Moreover, the first derivatives of the right-hand sides upper-bound those of the left-hand sides at time $t = 0$. However, this property at $t = 0$ does not immediately generalize to all $t > 0$. In our proof, we first construct a strict upper bound of the left-hand sides of Equation (3.8) that holds for all t, and then show that this bound converges to $\mu(\cdot)$.

First, for any positive $\epsilon > 0$, we construct a pair of ϵ-factor trajectories $(\mu_{1\epsilon}, \mu_{2\epsilon})$ with derivatives ϵ-close to the trajectory of μ in Equation (3.7) and show that these trajectories strictly upper-bound the discrepancy functions of V_1 and V_2.

For any $\delta_1, \delta_2 \geq 0$ and any $\epsilon > 0$, a pair of trajectories $\mu_{1\epsilon}, \mu_{2\epsilon} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ are defined as solutions to the differential equations:

$$\begin{aligned}\dot{\mu}_{1\epsilon}(t) &= \dot{\beta}_1(\delta_1, t) + \gamma_1 \underline{\alpha}_2^{-1}(\mu_{2\epsilon}(t - d(1,2))) + \epsilon, \text{ and} \\ \dot{\mu}_{2\epsilon}(t) &= \dot{\beta}_2(\delta_2, t) + \gamma_2 \underline{\alpha}_1^{-1}(\mu_{1\epsilon}(t - d(2,1))) + \epsilon,\end{aligned} \tag{3.9}$$

with $\mu_{1\epsilon}(0) = \beta_1(\delta_1, 0) + \epsilon$ and $\mu_{2\epsilon}(0) = \beta_2(\delta_2, 0) + \epsilon$. The right-hand side of Equation (3.9) is Lipschitz, and therefore, the solutions $\mu_{1\epsilon}$ and $\mu_{2\epsilon}$ are well-defined differentiable functions of time. For any two initial states of $\mathcal{A}$, θ, θ', we define two differentiable functions $g_1, g_2 : \mathbb{R}_{\geq 0} \to \mathbb{R}$:

$$\begin{aligned}g_1(t) = \mu_{1\epsilon}(t) - \beta_1(\delta_1, t) - \textstyle\int_0^t \gamma_1(|\xi_2(s - d(1,2)) - \xi_2'(s - d(1,2))|)ds, \\ g_2(t) = \mu_{2\epsilon}(t) - \beta_2(\delta_2, t) - \textstyle\int_0^t \gamma_2(|\xi_1(s - d(2,1)) - \xi_1'(s - d(2,1))|)ds.\end{aligned} \tag{3.10}$$

Recall that $\xi = Traj(\mathcal{A}, \theta)$ and $\xi' = Traj(\mathcal{A}, \theta')$ are the trajectories of $\mathcal{A}$ starting from θ and θ'. The following proposition states that if $g_1(t), g_2(t)$ are positive for all $t \geq 0$, then the distance between trajectories ξ and ξ' are bounded by $(\mu_{1\epsilon}, \mu_{2\epsilon})$.

Proposition 3.4. *Consider any non-negative pair $\delta = (\delta_1, \delta_2)$ and initial*

states $\theta, \theta' \in \Theta_{\mathcal{A}}$ *such that* $\theta' \in \mathcal{B}_\delta(\theta)$. *Let* $\xi = Traj(\mathcal{A}, \theta)$ *and* $\xi' = Traj(\mathcal{A}, \theta')$. *Then, for any* $\epsilon > 0, t \geq 0$, *if* $g_1(t), g_2(t) > 0$, *then*

$$V_1(\xi_1(t), \xi_1'(t)) < \mu_{1\epsilon}(t), \text{ and } V_2(\xi_2(t), \xi_2'(t)) < \mu_{2\epsilon}(t).$$

Proof. Here we prove the bound for V_1; the bound for V_2 follows by symmetry. For any $t \geq 0$, since $g_1(t) > 0$, from Equatoin (3.10) we have

$$\mu_{1\epsilon}(t) > \beta_1(\delta_1, t) + \int_0^t \gamma_1(|\xi_2(s - d(1,2)) - \xi_2'(s - d(1,2))|)ds. \tag{3.11}$$

From $\theta' \in \mathcal{B}_\delta(\theta)$, we have $|\theta_1 - \theta_1'| \leq \delta_1$. Since $\beta_1(\cdot, t)$ is a class-$\mathcal{K}$ function, it follows that

$$\beta_1(\delta_1, t) \geq \beta_1(|\theta_1 - \theta_1'|, t).$$

Thus, Equation (3.11) becomes

$$\mu_{1\epsilon}(t) > \beta_1(|\theta_1 - \theta_1'|, t) + \int_0^t \gamma_1(|\xi_2(s - d(1,2)) - \xi_2'(s - d(1,2))|)ds.$$

By applying Equation (3.6), it follows that

$$\begin{aligned} \mu_{1\epsilon}(t) &> \beta_1(|\theta_1 - \theta_1'|, t) + \textstyle\int_0^t \gamma_1(|\xi_2(s - d(1,2)) - \xi_2'(s - d(1,2))|)ds \\ &\geq V_1(\xi_1(t), \xi_1'(t)). \end{aligned}$$

□

The next lemma establishes that we can drop the assumption about the positivity of g_1 and g_2 and still arrive at the conclusion of Proposition 3.4.

Lemma 3.5. *Consider any non-negative pair* $\delta = (\delta_1, \delta_2)$, *and initial states* $\theta, \theta' \in \Theta_A$ *such that* $\theta' \in \mathcal{B}_\delta(\theta)$. *Let* $\xi = Traj(\mathcal{A}, \theta)$ *and* $\xi' = Traj(\mathcal{A}, \theta')$. *Then, for any* $\epsilon > 0, t \geq 0$,

$$V_1(\xi_1(t), \xi_1'(t)) < \mu_{1\epsilon}(t) \text{ and } V_2(\xi_2(t), \xi_2'(t)) < \mu_{2\epsilon}(t).$$

Proof. By Proposition 3.4, it suffices to prove that for all $t \geq 0$, $g_1(t), g_2(t) > 0$. At $t = 0$, by Equation (3.10)

$$g_1(0) = \beta_1(\delta_1, 0) + \epsilon - \beta_1(\delta_1, 0) = \epsilon > 0.$$

Similarly, $g_2(0) > 0$. Suppose for the sake of contradiction that $t_a > 0$ is the first time when $g_1(t), g_2(t) > 0$ is violated. From the continuity of g_1, g_2, we have that the both of the following conditions hold:

(i) $g_1(t), g_2(t) > 0$ for all $t \in [0, t_a)$, and

(ii) $g_1(t_a) = 0$ or $g_2(t_a) = 0$.

Without loss of generality, we assume $g_1(t_a) = 0$. From the mean value theorem, we know that there exists some time $t_b \in (0, t_a)$ such that

$$\dot{g}_1(t_b) = \frac{g_1(0) - g_1(t_a)}{0 - t_a} \leq -\frac{\epsilon}{t_a} < 0. \tag{3.12}$$

We can bound the derivative $\dot{g}_1(t_b)$ as:

$$\dot{g}_1(t_b) = \dot{\mu}_{1\epsilon}(t_b) - \frac{d}{dt}\left(\beta_1(\delta_1, t_b) + \int_0^{t_b} \gamma_1(|\xi_2(s - d(1,2)) - \xi_2'(s - d(1,2))|)ds\right).$$

Plugging the right-hand side of Equation (3.9) into the above equation, it follows:

$$\begin{aligned}
\dot{g}_1(t_b) = \;& \dot{\beta}_1(\delta_1, t_b) + \gamma_1 \underline{\alpha}_2^{-1}(\mu_{2\epsilon}(t_b - d(1,2))) + \epsilon \\
& -\dot{\beta}_1(\delta_1, t_b) - \gamma_1(|\xi_2(t_b - d(1,2)) - \xi_2'(t_b - d(1,2))|) \\
= \;& \epsilon + \gamma_1 \underline{\alpha}_2^{-1}(\mu_{2\epsilon}(t_b - d(1,2))) - \gamma_1(|\xi_2(t_b - d(1,2)) - \xi_2'(t_b - d(1,2))|).
\end{aligned} \tag{3.13}$$

From condition (i), we know $g_2(t_b - d(1,2)) > 0$. It follows from Proposition 3.4 that

$$\mu_{2\epsilon}(t_b - d(1,2)) > V_2(\xi_2(t_b - d(1,2)), \xi_2'(t_b - d(1,2))). \tag{3.14}$$

From Definition 3.4, we have $V_2(\xi_2(t_b), \xi_2'(t_b)) \geq \underline{\alpha}_2(|\xi_2(t_b - d(1,2)) - \xi_2'(t_b - d(1,2))|)$. Equation (3.14) yields $\mu_{2\epsilon}(t_b - d(1,2)) > \underline{\alpha}_2(|\xi_2(t_b - d(1,2)) - \xi_2'(t_b - d(1,2))|)$. Because $\gamma \underline{\alpha}_2^{-1}$ is a class-$\mathcal{K}$ function, it follows that

$$\gamma_1 \underline{\alpha}_2^{-1}(\mu_{2\epsilon}(t_b - d(1,2))) \geq \gamma_1(|\xi_2(t_b - d(1,2)) - \xi_2'(t_b - d(1,2))|).$$

Combining the above equation with Equation (3.13), we have that $\dot{g}_1(t_b) \geq \epsilon > 0$, which contradicts Equation (3.12). □

Lemma 3.5 shows that for any $\epsilon > 0$, the ϵ-factor trajectories $\mu_{1\epsilon}$ and $\mu_{2\epsilon}$ give strict upper bounds on the distance between corresponding trajectories of the original modules $\mathcal{A}_1$ and $\mathcal{A}_2$. In the following lemma, we show that as $\epsilon \to 0$, $\mu_{i\epsilon}$ converges to the trajectory $\mu \downarrow m_i$; recall that μ is the trajectory of the IS approximation M. It follows that the trajectory μ indeed bounds the divergence of any trajectories of $\mathcal{A}$.

Lemma 3.6. *Consider any non-negative pair $\delta = (\delta_1, \delta_2)$ and initial states $\theta, \theta' \in \Theta_{\mathcal{A}}$ such that $\theta' \in \mathcal{B}_\delta(\theta)$. Let $\xi = Traj((\mathcal{A}, \theta))$, $\xi' = Traj((\mathcal{A}, \theta'))$, and μ be the trajectory of $\mathcal{A}$'s (δ_1, δ_2)-IS approximation M. Then for all $t \geq 0$,*

$$(\xi(t), \xi'(t)) \in L_V(\mu(t)).$$

Proof. For brevity we write $\mu \downarrow m_i$ as μ_i. By integrating both sides of Equation (3.7) with initial condition $\mu_1(0) = \beta_1(\delta_1, 0)$ and $\mu_2(0) = \beta_2(\delta_2, 0)$, we have,

$$\begin{aligned} \mu_1(t) = &\quad \beta_1(\delta_1, t) + \textstyle\int_0^t \gamma_1 \underline{\alpha}_2^{-1}(\mu_2(s - d(1,2)))ds, \\ \mu_2(t) = &\quad \beta_2(\delta_2, t) + \textstyle\int_0^t \gamma_2 \underline{\alpha}_1^{-1}(\mu_1(s - d(2,1)))ds. \end{aligned} \tag{3.15}$$

Similarly, by integrating Equation (3.9), we have

$$\begin{aligned} \mu_{1\epsilon}(t) = &\quad \beta_1(\delta_1, t) + \textstyle\int_0^t \gamma_1 \underline{\alpha}_2^{-1}(\mu_{2\epsilon}(s - d(1,2)))ds + \epsilon(1+t), \\ \mu_{2\epsilon}(t) = &\quad \beta_2(\delta_1, t) + \textstyle\int_0^t \gamma_2 \underline{\alpha}_1^{-1}(\mu_{1\epsilon}(s - d(2,1)))ds + \epsilon(1+t). \end{aligned} \tag{3.16}$$

Define $h(t) \triangleq |\mu_1(t) - \mu_{1\epsilon}(t)| + |\mu_2(t) - \mu_{2\epsilon}(t)|$. Plugging in Equations (3.15) and (3.16) into the definition of $h(t)$, we have,

$$\begin{aligned} h(t) \quad \leq \quad & 2\epsilon(t+1) + \textstyle\int_0^t |\gamma_1 \underline{\alpha}_2^{-1}(\mu_{1\epsilon}(s - d(1,2))) - \gamma_1 \underline{\alpha}_2^{-1}(\mu_1(s - d(1,2)))|ds \\ & + \textstyle\int_0^t |\gamma_2 \underline{\alpha}_1^{-1}(\mu_{2\epsilon}(s - d(2,1))) - \gamma_2 \underline{\alpha}_1^{-1}(\mu_2(s - d(2,1)))|ds. \end{aligned} \tag{3.17}$$

We will bound the two integrals of above inequality in Equations (3.18) and (3.19). The property of delay function implies that for any $t \in [0, d]$,

$\mu(t-d) = \mu(0)$. By splitting the integral intervals, we have,

$$\begin{aligned} & \int_0^t |\gamma_1 \underline{\alpha}_2^{-1}(\mu_{1\epsilon}(s - d(1,2))) - \gamma_1 \underline{\alpha}_2^{-1}(\mu_1(s - d(1,2)))| ds \\ = \; & \int_0^{d(1,2)} |\gamma_1 \underline{\alpha}_2^{-1}(\mu_{1\epsilon}(0)) - \gamma_1 \underline{\alpha}_2^{-1}(\mu_1(0))| ds \\ & + \int_0^{t-d(1,2)} |\gamma_1 \underline{\alpha}_2^{-1}(\mu_{1\epsilon}(s)) - \gamma_1 \underline{\alpha}_2^{-1}(\mu_1(s))| ds \\ \leq \; & d(1,2) |\gamma_1 \underline{\alpha}_2^{-1}(\mu_{1\epsilon}(0)) - \gamma_1 \underline{\alpha}_2^{-1}(\mu_1(0))| \\ & + \int_0^t |\gamma_1 \underline{\alpha}_2^{-1}(\mu_{1\epsilon}(s)) - \gamma_1 \underline{\alpha}_2^{-1}(\mu_1(s))| ds. \end{aligned} \tag{3.18}$$

By using the same trick, we rewrite the last term of Equation (3.17) as

$$\begin{aligned} & \int_0^t |\gamma_2 \underline{\alpha}_1^{-1}(\mu_{2\epsilon}(s - d(2,1))) - \gamma_2 \underline{\alpha}_1^{-1}(\mu_2(s - d(2,1)))| ds \\ \leq \; & d(2,1) |\gamma_2 \underline{\alpha}_1^{-1}(\mu_{2\epsilon}(0)) - \gamma_2 \underline{\alpha}_1^{-1}(\mu_2(0))| ds \\ & + \int_0^t |\gamma_2 \underline{\alpha}_1^{-1}(\mu_{2\epsilon}(s)) - \gamma_2 \underline{\alpha}_1^{-1}(\mu_2(s))| ds. \end{aligned} \tag{3.19}$$

From the Lipschitz property of $\gamma_1 \underline{\alpha}_2^{-1}$ and $\gamma_2 \underline{\alpha}_1^{-1}$, we can find a constant $L > 0$ such that $|\gamma_1 \underline{\alpha}_2^{-1}(\mu_{1\epsilon}(t)) - \gamma_1 \underline{\alpha}_2^{-1}(\mu_1(t))| \leq L|\mu_1(t) - \mu_{1\epsilon}(t)|$ and $|\gamma_2 \underline{\alpha}_1^{-1}(\mu_{2\epsilon}(t)) - \gamma_2 \underline{\alpha}_1^{-1}(\mu_2(t))| \leq L|\mu_2(t) - \mu_{2\epsilon}(t)|$. Using the Lipschitz condition with Equations (3.18) and (3.19), Equation (3.17) becomes

$$h(t) \leq 2\epsilon(t + 1 + L d_M) + \int_0^t L h(s) ds,$$

with $d_M = \max\{d(1,2), d(2,1)\}$. By Gronwall-Bellman inequality [70], it follows that

$$h(t) \leq 2\epsilon(t + 1 + L d_M) + 2\epsilon L \int_0^t (s + 1 + L d_M) e^{L(t-s)} ds. \tag{3.20}$$

It follows that for any $t \in \mathbb{R}_{\geq 0}$, the integral $\int_0^t (s+1) e^{L(t-s)} ds$ is bounded. Thus $h(t) \to 0$ as $\epsilon \to 0$, which implies both $|\mu_1(t) - \mu_{1\epsilon}(t)| \to 0$ and $|\mu_2(t) - \mu_{2\epsilon}(t)| \to 0$ as $\epsilon \to 0$. Using Lemma 3.5, and taking the limit of $\epsilon \to 0$, it follows that

$$V_1(\xi_1(t), \xi_1'(t)) \leq \mu_1(t) \text{ and } V_2(\xi_2(t), \xi_2'(t)) \leq \mu_2(t).$$

That is, for any $t \geq 0$, $(\xi(t), \xi'(t)) \in L_V(\mu(t))$. □

Theorem 3.7 states that the reach set of any (large) networked dynamical system $\mathcal{A} = \|_d\{\mathcal{A}_1, \mathcal{A}_2\}$ from a set of states can be over-approximated by bloating an individual execution ξ of $\mathcal{A}$ by a factor that is entirely determined by (a) IS discrepancy functions V_1 and V_2 of its (smaller) modules, and (b) the trajectory μ of the reduced (two-dimensional) dynamical system M that is its IS approximation.

Theorem 3.7. *Consider a networked dynamical system $\mathcal{A} = \|_d\{\mathcal{A}_1, \mathcal{A}_2\}$ with IS discrepancy functions $V = (V_1, V_2)$. Let $\xi = Traj(\mathcal{A}, \theta)$ for some initial state $\theta \in \Theta_{\mathcal{A}}$. For any non-negative pair $\delta = (\delta_1, \delta_2)$ suppose μ is the trajectory of the (δ_1, δ_2)-IS approximation M. Then, for any $T \geq 0$*

$$\mathsf{Reach}_{\mathcal{A}}(\mathcal{B}_\delta(\theta), [0, T]) \subseteq \bigcup_{t \in [0,T]} \mathcal{B}^V_{\mu(t)}(\xi(t)).$$

Proof. This follows from the bounds on the distance between trajectories established above. For any $\mathbf{x} \in \mathsf{Reach}_{\mathcal{A}}(\mathcal{B}_\delta(\theta), [0, T])$, there exists an initial state $\theta' \in \mathcal{B}_\delta(\theta)$, a trajectory $\xi' = Traj(\mathcal{A}, \theta')$ and a time $t \in [0, T]$ such that $\xi'(t) = \mathbf{x}$. It follows by Lemma 3.6 that $(\xi(t), \xi'(t)) \in L_V(\mu(t))$, and therefore, $\mathbf{x} \in \mathcal{B}^V_{\mu(t)}(\xi(t))$. □

Theorem 3.7 establishes an over-approximation of the set of reachable states from a δ-ball $\mathcal{B}_\delta(\theta)$. To compute the set of reachable states from a compact initial set Θ_A, we can proceed as usual by first computing a δ-cover of Θ_A, and then computing the union of reach sets of the covers using the theorem.

Theorem 3.7 does not require $\mathcal{A}$ to be stable or any global property to hold for the IS discrepancy functions. To use Theorem 3.7 we need to (a) find IS discrepancy functions for the modules, and (b) compute individual trajectories ξ of complete network $\mathcal{A}$ and μ of M. Fortunately, for large classes of nonlinear dynamical systems there exist scalable numerical techniques for (b). This is one of the motivations of this work. For linear and some special classes of nonlinear systems (a) can be solved automatically (see Section 3.4.2). For nonlinear models, we refer the readers to [61], where we present an algorithmic approach for computing local versions of the discrepancy function.

3.5.3 Precision of the IS Approximation

The error in the over-approximation obtained from the reduced model is determined by the trajectories of the reduced model. In the following, we show that the over-approximation error can be made arbitrarily small with sufficiently small but positive δ_1, δ_2.

Lemma 3.8. *Consider any $T > 0$, $t \in [0, T]$, and a sequence of pairs of positive reals $\delta^k = (\delta_1^k, \delta_2^k)$ converging to $(0, 0)$. For the trajectory (μ^k) of the corresponding (δ_1^k, δ_2^k)-IS approximation M^k, $|(\mu^k \downarrow m_i)(t)| \to 0$ for $i \in \{1, 2\}$.*

Proof. Fix a $T > 0$ and $\delta^k = (\delta_1^k, \delta_2^k)$. This defines the (δ_1^k, δ_2^k)-IS approximation M^k and its trajectory μ^k. We will prove that for all $t \in [0, T]$,

$$|(\mu^k \downarrow m_1)(t)| + |(\mu^k \downarrow m_2)(t)| \to 0,$$

as $\delta^k \to 0$. From here on, we drop the superscript k and use the notations setup earlier: $\mu_i = \mu \downarrow m_i$, etc.

From the first row in Equation (3.15), applying the same trick as Equation (3.18), we have that

$$|\mu_1(t)| \leq \beta_1(\delta_1, t) + d(1,2)|\gamma_1 \underline{\alpha}_2^{-1}(\mu_2(0))| + \int_0^t |\gamma_1 \underline{\alpha}_2^{-1}(\mu_2(s))| ds. \tag{3.21}$$

From the Lipschitz property of $\gamma_1 \underline{\alpha}_2^{-1}$, there exists some $L > 0$, such that $|\gamma_1 \underline{\alpha}_2^{-1}(\mu_2(s)) - \gamma_1 \underline{\alpha}_2^{-1}(0)| \leq L|\mu_2(s) - 0|$. Since $\gamma_1 \underline{\alpha}_2^{-1}$ is of class-$\mathcal{K}$, it follows that

$$|\gamma_1 \underline{\alpha}_2^{-1}(\mu_2(s))| \leq L|\mu_2(s)|.$$

From Equation (3.15), we observe that for $i \in \{1, 2\}$, $\mu_i(t)$ are non-negative scalars. We drop the absolute value symbols $|\cdot|$. Then Equation (3.21) reduces to

$$|\mu_1(t)| \leq \beta_1(\delta_1, t) + d(1,2) L \beta_1(\delta_1, 0) + \int_0^t L|\mu_2(s)| ds. \tag{3.22}$$

Since $\beta_1(\delta_1, t)$ is bounded in compact intervals, we define

$$C_1^T(\delta_1) = \sup_{t \in [0,T]} \beta_1(\delta_1, t), \tag{3.23}$$

as the upper bound of the function $\beta_1(\cdot, t)$ in interval $t \in [0, T]$. It follows from Equation (3.22) that

$$|\mu_1(t)| \leq C_1^T(\delta_1) + d(1,2)L\beta_1(\delta_1, 0) + \int_0^t L|\mu_2(s)|ds. \tag{3.24}$$

Starting from the second row of Equation (3.15), and following similar steps from Equations (3.21)-(3.24), we have,

$$|\mu_2(t)| \leq C_2^T(\delta_2) + d(2,1)L\beta_2(\delta_2, 0) + \int_0^t L|\mu_1(s)|ds. \tag{3.25}$$

Summing up Equations (3.24) and (3.25), by applying the Gronwall-Bellman inequality, we have

$$|\mu_1(t)| + |\mu_2(t)| \leq (C_1^T(\delta_1) + C_2^T(\delta_2) + d(1,2)L\beta_1(\delta_1, 0) + d(2,1)L\beta_2(\delta_2, 0))e^{Lt}.$$

For $i \in \{1, 2\}$, $\beta_i(\cdot, \cdot)$ is a class-$\mathcal{K}$ function in the first argument, thus $\beta_i(\delta_i^k, 0) \to 0$ and $C_i(\delta_i^k) \to 0$ as $\delta_i^k \to 0$. It follows that, $|\mu_1^k(t)| + |\mu_2^k(t)| \to 0$ as $\delta^k \to 0$. □

Proposition 3.9 follows from the fact that for $i \in \{1, 2\}$, for any $\mathbf{x}, \mathbf{x}' \in X_i$, $\underline{\alpha}_i(\mathbf{x} - \mathbf{x}') \leq V_i(\mathbf{x}, \mathbf{x}')$ (Definition 3.4).

Proposition 3.9. *For dynamical system $\mathcal{A}$ with discrepancy function $V = (V_1, V_2)$, fix any $\mathbf{x} \in X_{\mathcal{A}}$. For any $\epsilon > 0$, there exists $r > 0$, such that $\mathcal{B}_r^V(\mathbf{x}) \subseteq \mathcal{B}_\epsilon(\mathbf{x})$.*

Using Lemma 3.8 and Proposition 3.9, the next theorem bounding the error on the over-approximation of the reach set follows in a straightforward fashion.

Theorem 3.10. *Consider a closed dynamical system $\mathcal{A} = ||_d\{\mathcal{A}_1, \mathcal{A}_2\}$ with IS discrepancy $V = (V_1, V_2)$. Let $\xi = Traj(\mathcal{A}, \theta)$ for some initial state $\theta \in \Theta_{\mathcal{A}}$. For any $\epsilon > 0$, there exists a positive pair $\delta_1, \delta_2 > 0$, such that for $\mathcal{A}$'s (δ_1, δ_2)-IS approximation M, for any $T \geq 0$,*

$$\bigcup_{t \in [0,T]} \mathcal{B}_{\mu(t)}^V(\xi(t)) \subseteq \mathcal{B}_\epsilon(\mathsf{Reach}_{\mathcal{A}}(\mathcal{B}_\delta(\theta), [0, T])),$$

where μ is the unique trajectory of M.

If the over-approximation obtained from Theorem 3.7 is not precise enough to prove safety, then, we can refine the parameters δ_1 and δ_2. Then we can compute $\mathcal{B}^V_{\mu(t)}(\xi(t))$ for each of the smaller partitions with higher precision. This is the standard approach used in simulation-based verification [52, 31, 71]. In the next section, we present this algorithm in more detail.

3.5.4 Generalizing to Arbitrary Networked Dynamical Systems

The approximation and its analysis presented in the previous section can be extended to closed networked dynamical systems. Let $\{\mathcal{A}_i\}_{i\in[N]}$ be a compatible closed collection of dynamic modules and $d \in {\mathbb{R}_{\geq 0}}^{N\times N}$ be a delay matrix. Let $\mathcal{A} = ||_d\{\mathcal{A}_i\}_{i\in[N]}$ be the corresponding closed dynamic network. For any pair of trajectories $\xi, \xi' \in Traj(\mathcal{A}, \Theta)$, and for each $i \in [N]$, we define $\xi_i = \xi \downarrow X_i$ and $\xi'_i = \xi' \downarrow X_i$. The next definition gives a natural generalization of Definition 3.4 to account for dynamical modules that take inputs from more than one module in the network.

Definition 3.7. *Let $\mathcal{A}_i = \langle X_i, U_i, \theta_i, f_i\rangle$ be a dynamic module receiving input signals from a collection of modules $\{\mathcal{A}_j\}_{j\in U_i\cap X_j}$. An input-to-state discrepancy function for $\mathcal{A}_i$ (in this environment) is a continuous function $V_i : Val(X)^2 \to \mathbb{R}_{\geq 0}$ with class-$\mathcal{K}$ witnesses $(\underline{\alpha}_i, \overline{\alpha}_i, \beta_i)$ and $\{\gamma_{ij}\}_{(i,j):U_i\cap X_j\neq\emptyset}$, such that*

(i) for any $\mathbf{x}, \mathbf{x}' \in Val(X_i)$, $\underline{\alpha}_i(|\mathbf{x} - \mathbf{x}'|) \leq V_i(\mathbf{x}, \mathbf{x}') \leq \overline{\alpha}_i(|\mathbf{x} - \mathbf{x}'|)$, and

(ii) for any pair of initial states $\theta, \theta' \in \Theta_i$ and input signals $\eta_{ij}, \eta'_{ij} \in Traj(U_i \cap X_j)$ from module $\mathcal{A}_j$ to $\mathcal{A}_i$, for each $t \geq 0$

$$V_i(\xi_i(t), \xi'_i(t)) \leq \beta_i(|\theta - \theta'|, t) + \int_0^t \sum_{j:X_j\cap U_i\neq\emptyset} \gamma_{ij}(|\eta_{ij}(s) - \eta'_{ij}(s)|)ds,$$

where ξ_i and ξ'_i are the trajectories of $\mathcal{A}_i$ from initial state θ (θ' respectively) and with input signal η_{ij} (η'_{ij} respectively) from module $\mathcal{A}_j$.

The set $\{\mathcal{A}_j | X_j \cap U_i \neq \emptyset\}$ is the set of modules that provide inputs to module $\mathcal{A}_i$. Definition 3.7 allows each $\mathcal{A}_j$ providing an input to $\mathcal{A}_i$ to have a different witness γ_{ij} to the discrepancy function V_i. This flexibility also

permits the input signals from different modules to experience delays to different degrees. Generalizing Definition 3.6, the IS approximation of $\mathcal{A}$ is a N-dimensional closed deterministic dynamical system M.

Definition 3.8. *For any $\delta = (\delta_1, \ldots, \delta_N) \in \mathbb{R}^N_{\geq 0}$, the δ-IS approximation of $\mathcal{A} = ||_d\{\mathcal{A}_i\}_{i\in[N]}$ is a dynamical network $M = \langle X_M, \Theta_M, \mathcal{T}_M\rangle$, where*

(i) $X_M = \{m_1, m_2, \ldots, m_N\}$,

(ii) $\Theta_M = \{\theta\}$, *where* $\theta \lceil m_i = \beta_i(\delta_i, 0)$, *for each* $i \in [N]$, *and*

(iii) for any trajectory μ of X_M, $\mu \in \mathcal{T}_M$ if and only if

$$\dot{\mu}_i(t) = \dot{\beta}_i(\delta_i, t) + \sum_{j:X_j\cap U_i\neq\emptyset} \gamma_{ij}\underline{\alpha}_j^{-1}(\mu_j(t - d(i,j))),$$

where $\mu_i = \mu \downarrow m_i$.

The dynamics of the closed reduced network M is defined syntactically in terms of the IS discrepancy functions of the constituent modules $\{\mathcal{A}_i\}_{i\in[N]}$ and the delay matrix d. Once again, from the Lipschitz continuity of the right-hand side it follows that the delay differential equation has a unique solution. The IS approximation M of the original dynamical network $\mathcal{A}$ is an N-dimensional system. The construction of M is defined syntactically only using (a) information of individual modules (IS discrepancy functions etc.) and (b) the topology and delay matrix for composition. An N-dimensional analogue of Theorems 3.7 and 3.10 giving approximations of $\mathsf{Reach}_A(\Theta, [0, T])$ in terms of $\mu(\cdot)$ can be proven essentially following the same steps.

3.6 Verification Algorithm

Theorem 3.7 gives us a recipe for verifying bounded time invariants (or safety properties) of closed networks, only by numerically computing trajectories of the whole network $\mathcal{A}$ and the trajectories of the statically computed reduced IS approximation M. That is, detailed static analysis of the complete model $\mathcal{A}$ becomes unnecessary in this method. In this section, we develop this idea further and provide the details of the verification algorithm.

The algorithm involves simulations or numerical computation of trajectories of dynamical systems. We proceed by first defining what we mean

by simulations. Given a closed networked dynamical system $\mathcal{A}$ without delay ($d(i,j) = 0$ for all i,j), an initial state θ, let $\xi = Traj(\mathcal{A},\theta)$ be the actual trajectory of $\mathcal{A}$ from the initial state θ. For a step size $\tau > 0$, validated ODE solvers such as [72, 73, 74] compute a sequence of boxes $R_1, R_2, \ldots, R_l \subseteq Val(X_A)$, such that for each $k \in [l]$, $t \in [(k-1)\tau, k\tau]$, $\xi(t) \in R_k$. For a desired error bound $\epsilon > 0$, by reducing the step size τ, the diameter of R_k can be made smaller than ϵ. For delayed differential equations, there are existing techniques for numerical simulation [62, 75, 76, 77]. Several of these reduce the problem to simulating ODEs [78]. Specifically, the simulations generated in [62] have validated error bounds. Our safety verification algorithm assumes the availability of a subroutine *Simulate* for computing numerical simulations that meet Definition 3.9.

Definition 3.9. *Consider a closed network $\mathcal{A}$, an initial state θ, a time bound T, an error bound $\epsilon > 0$, and time step $\tau > 0$. Let $\xi = Traj(\mathcal{A},\theta)$ denote the trajectory of $\mathcal{A}$ from the initial state θ. A* $(\theta, T, \epsilon, \tau)$-simulation trace *is a finite sequence $\phi = \langle R_0, t_0\rangle, \langle R_1, t_1\rangle, \ldots, \langle R_l, t_l\rangle$ where*

(i) $t_0 = 0$, $t_l = T$, *and* $\forall\ k \in [l]$, $t_k - t_{k-1} \leq \tau$,

(ii) $\forall k \in [l]$ *and* $\forall t \in [t_{k-1}, t_k]$, $\xi(t) \in R_k$, *and*

(iii) $\forall k \in [l]$, $dia(R_k) \leq \epsilon$.

In Algorithm 3.1 the subroutine $Simulate(\mathcal{A}, \theta, T, \epsilon, \tau)$ (line 5,20) computes a $(\theta, T, \epsilon, \tau)$-simulation of the network $\mathcal{A}$ as defined above. For the completeness of the algorithm, it is required that for any precision parameters $\epsilon, \tau > 0$, a simulation trace fulfilling this condition can be computed.

The algorithm takes several inputs: (a) the system model $\mathcal{A} = \|_d\{\mathcal{A}_i\}_{i\in[N]}$, (b) the unsafe set $Unsafe \subseteq Val(X_{\mathcal{A}})$, (c) initial partition parameter δ_0, simulation parameter ϵ_0, and the time bound T, and (d) a collection of IS discrepancy functions and witnesses for the modules $\{V_i, \underline{\alpha}_i, \overline{\alpha}_i, \beta_i, \{\gamma_{ji}\}\}$.

The algorithm proceeds as follows: The set $\mathcal{C}$ computed in line 2 is a finite set of triples $\{(\theta_c, \delta, \epsilon)\}_{c\in|\mathcal{C}|}$, such that $\{\theta_c\}_{c\in|\mathcal{C}|}$ is a δ-cover of the initial set $\Theta_{\mathcal{A}}$. Each θ_c is associated with a precision parameter $\epsilon > 0$ and positive real-valued vector δ. For each triple $(\theta, \delta, \epsilon)$ in the cover set $\mathcal{C}$, the algorithm first computes a simulation trace $\psi = \langle R_0, t_0\rangle, \ldots, \langle R_p, t_p\rangle$ containing the trajectory $\xi = Traj(\mathcal{A},\theta)$ of the (large) networked dynamical system $\mathcal{A}$

Algorithm 3.1 Bounded verification algorithm

1: $\delta \leftarrow \delta_0; \epsilon \leftarrow \epsilon_0; \mathcal{R} \leftarrow \emptyset;$
2: $\mathcal{C} \leftarrow \mathit{Partition}(\Theta_{\mathcal{A}}, \delta, \epsilon);$
3: **while** $\mathcal{C} \neq \emptyset$ **do**
4: **for** $(\theta, \delta, \epsilon) \in \mathcal{C}$ **do**
5: $\psi \leftarrow \mathit{Simulate}(\mathcal{A}, \theta, T, \epsilon, \tau);$
6: $S \leftarrow \mathit{BloatWithISD}(\psi, d, \delta, \epsilon, \tau, \{V_i, \underline{\alpha}_i, \overline{\alpha}_i, \beta_i, \{\gamma_{ij}\}\});$
7: **if** $S \cap \mathit{Unsafe} = \emptyset$ **then**
8: $\mathcal{C} \leftarrow \mathcal{C} \backslash \{(\theta, \delta, \epsilon)\}; \mathcal{R} \leftarrow \mathcal{R} \cup S$;
9: **else if** $\exists k, R_k \subseteq \mathit{Unsafe}$ **then**
10: **return** (UNSAFE, $\mathcal{R}$);
11: **else**
12: $\mathcal{C} \leftarrow \mathcal{C} \backslash \{(\theta, \delta, \epsilon)\};$
13: $\mathcal{C} \leftarrow \mathcal{C} \cup \mathit{Partition}(\Theta_{\mathcal{A}} \cap \mathcal{B}_\delta(\theta), \frac{\delta}{2}, \frac{\epsilon}{2})$
14: **end if**
15: **end for**
16: **end while**
17: **return** (SAFE, $\mathcal{R}$);
18: **function** $\mathit{BloatWithISD}(\psi, d, \delta, \epsilon, \tau, \{V_i, \underline{\alpha}_i, \overline{\alpha}_i, \beta_i, \{\gamma_{ij}\}\})$
19: $M \leftarrow \mathit{ISApprox}(\delta, d, \{V_i, \underline{\alpha}_i, \overline{\alpha}_i, \beta_i, \{\gamma_{ij}\}\});$
20: $\phi \leftarrow \mathit{Simulate}(M, \theta_M, T, \epsilon, \tau);$
21: $\rho \leftarrow \mathit{SupByTime}(\phi);$
22: $S \leftarrow \mathcal{B}^V_\rho(\psi);$
23: **return** S;
24: **end function**

(line 5). Then the subroutine *BloatWithISD* bloats the trace ψ to get a tube S (line 6). We claim that the tube S contains the set of states of $\mathcal{A}$ reachable from the set of initial state $\mathcal{B}_\delta(\theta)$. Then the algorithm checks whether this tube is safe or unsafe. If neither of these cases can be deduced then the part of the initial set $\mathcal{B}_\delta(\theta)$ is further refined by adding a new partition to $\mathcal{C}$.

The subroutine *BloatWithISD* is detailed in lines 18-24. First, an δ-IS approximation M (line 19) is constructed following Definition 3.8. The *Simulate* subroutine is then used again to compute a simulation trace $\phi = \langle Q_0, t_0\rangle, \ldots, \langle Q_p, t_p\rangle$ of the trajectory $\mu = Traj(M, \theta_M)$, of the (smaller) reduced model M. Here we assume the time stamps of the sequence ψ match up with that of ϕ. This can be achieved by using a fixed step solver or through repeated simulations using smaller step sizes. The sequence ρ computed in line 21 is a sequence of pairs $\langle r_0, t_0\rangle, \ldots, \langle r_p, t_p\rangle$, where for each $k \in [p]$, r_k is a non-negative real defined as $r_k = \sup_{\mathbf{m} \in Q_k} |\mathbf{m}|$. In line 22, the sequence of

sets $\{R_k\}$ is bloated by the sequence of factors $\{r_k\}$ element-wise to construct a tube S.

3.6.1 Analysis of the Verification Algorithm

We establish the soundness and relative completeness of the algorithm in Theorems 3.11 and 3.12. Practical scalability of the algorithm is discussed in Section 3.7.

Theorem 3.11. *Algorithm 3.1 is sound. That is, if it returns* SAFE *then $\mathcal{A}$ is safe up to T, and if it returns* UNSAFE *then $\mathcal{A}$ is unsafe.*

Proof. Suppose the algorithm returns SAFE. For any cover $(\theta, \delta, \epsilon) \in \mathcal{C}$, S computed in line 22 is the union of a sequence of regions $\{\mathcal{B}^V_{r_k}(R_k)\}$, where $\cup_{k\in[p]} R_k$ contains the trajectory from θ and the sequence of r_k upper bounds the trajectory of δ-IS approximation. It follows from Theorem 3.7, that $\mathsf{Reach}_{\mathcal{A}}(\mathcal{B}_\delta(\theta), [0, T]) \subseteq S$. Thus if $S \cap \mathit{Unsafe} = \emptyset$, then $\mathsf{Reach}_{\mathcal{A}}(\mathcal{B}_\delta(\theta), [0, T]) \cap \mathit{Unsafe} = \emptyset$. The algorithm returns SAFE, if all such covers are checked safe. It is straightforward to check that $\mathcal{C}$ captures a finite cover of the initial set $\bigcup_{(\theta,\delta,\epsilon)\in\mathcal{C}} \mathcal{B}_\delta(\theta) \supseteq \Theta_{\mathcal{A}}$. Therefore, we conclude $\mathsf{Reach}_{\mathcal{A}}(\Theta_{\mathcal{A}}, [0, T]) \cap \mathit{Unsafe} = \emptyset$.

Otherwise if the algorithm returns UNSAFE, there exists at least one set R_k contained in the unsafe set *Unsafe*. It follows that for the trajectory $\xi = Traj(\mathcal{A}, \theta)$ and some $t \in [t_{k-1}, t_k]$, $\xi(t) \in \mathit{Unsafe}$. □

Theorem 3.12. *Algorithm 3.1 is relatively complete. That is, if $\mathcal{A}$ is robustly safe then Algorithm 3.1 terminates and returns* SAFE *and if $\mathcal{A}$ is unsafe then it terminates and returns* UNSAFE.

Proof. Suppose that for some $\epsilon' > 0$, $\mathcal{A}$ is ϵ'-robustly safe up to time T. It follows from Definition 3.2 that

$$\mathcal{B}_{\epsilon'}(\mathsf{Reach}_{\mathcal{A}}(\Theta_{\mathcal{A}}, [0, T])) \cap \mathit{Unsafe} = \emptyset.$$

It follows that line 8 is never executed. For any $\theta \in \Theta$, we will show that for small enough refinement parameters $\delta, \epsilon > 0$, for any k, $dia(\mathcal{B}^V_{r_k}(R_k)) < \epsilon'$. From Proposition 3.9, we can show that there exists $e > 0$ such that for $r_k < e$,

$$\mathcal{B}^V_{r_k}(R_k) \subseteq \mathcal{B}_{\epsilon'/3}(R_k). \tag{3.26}$$

From Lemma 3.8, there exists a $\delta > 0$, for all $t \in [0, T]$, and all $i \in [N]$, $|\mu(t)| \leq e/2$. For a simulation trace $\phi = \langle M_0, t_0 \rangle, \ldots, \langle M_q, t_q \rangle$ (line 20), and $\epsilon \leq e/2$, it follows from Definition 3.9 that for all $k \in [q]$, the diameter $dia(M_k) \leq e/2$. Thus for any $k \in [q]$, $r_k = \sup_{\mathbf{m} \in M_k} |\mathbf{m}| \leq \epsilon + \sup_{t \in [t'_{l-1}, t'_l]} |\mu(t)| \leq e/2 + e/2 = e$. It follows from Equation (3.26) that by choosing $\delta \leq d$ and $\epsilon \leq e/2$, we have $\mathcal{B}^V_{r_k}(R_k) \subseteq \mathcal{B}_{\epsilon'/3}(R_k)$. Notice that the diameter of R_k is bounded by the refinement parameter ϵ. By choosing $\epsilon = \min\{\epsilon'/3, e\}$, it follows that

$$dia(\mathcal{B}_{\epsilon'/3}(R_k)) \leq \epsilon'/3 + dia(R_k) \leq \epsilon'/3 + \epsilon'/3 < \epsilon'.$$

Thus, after a number of $\max\{\log(\frac{\epsilon_0}{\min\{\epsilon'/3, e\}}), \log \frac{\delta_0}{d}\}$ refinements, the parameters δ, ϵ are small enough to guarantee that $S \cap \mathit{Unsafe} = \emptyset$. Thus the algorithm returns SAFE.

On the other hand, suppose A is unsafe with respect to an open unsafe set Unsafe. There exists an initial state θ, a time $t \geq 0$ and a $\epsilon' > 0$ such that $\mathcal{B}_{\epsilon'}(\xi(\theta, t)) \subseteq \mathit{Unsafe}$. For the same number of refinement as the robustly safe case, it can be shown that there exists an $\mathcal{B}^V_{r_k}(R_k) \subseteq \mathcal{B}_{\epsilon'}(\xi(\theta, t))$. It follows that the algorithm returns UNSAFE. □

3.7 Experimental Validation

We discuss experimental results from a prototype implementation of Algorithm 3.1 in Matlab. We verify bounded time invariant properties of several linear and nonlinear networked dynamical systems. First, we give a brief overview of the different examples. Each module in the linear synchronization examples (see [79] for detail) is a four-dimensional linear dynamical system, and the overall system is composed of several modules in different topologies. Through communicating with neighbors, the modules in the network aim to reach synchronization to a common solution. We compute reach sets for several such networks, both with and without delays. The nonlinear water tank network example is a modified version of the one presented in [59]. In this example, each module captures the water levels in a group of tanks, that depends on the flows from other tanks. The nonlinear cardiac cell network with delayed interconnections was shown in Examples 3.1 and 3.2.

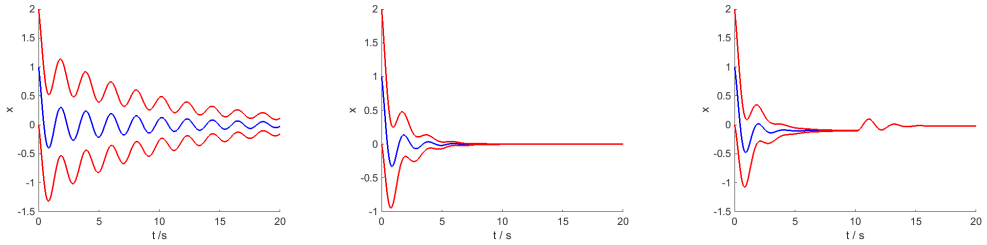
In the results presented here, the time bound is $T = 20$ seconds and the running time is based on executing the procedure on an Intel i5-3470 CPU. The columns in Table 3.1 present (i) the system being verified, (ii) the number of total state variables, (iii) the number of modules, (iv) the number of covers for the initial set, (v) the total number of simulation boxes generated, and (vi) the running time of the algorithm. Our experimental results show that the running time roughly scales linearly with the total number of simulation boxes generated for both the original system $\mathcal{A}$ and its IS approximation $\mathcal{M}$. The number of simulation boxes generated is proportional to the product of the total number of covers and the time bound. Fixing a compact initial set, the number of covers generated depends on the level of precision needed to prove (or disprove) safety, which further relying on the distance between the unsafe set to the reachable states. From the linear synchronization examples, we also observe that verifying time-delayed networks takes a longer time than delay-free networks of the same dimensions, even if the algorithm produces similar number of simulations boxes. One source of this difference is that simulating a delayed differential equation is more expensive than simulating an ODE.

Table 3.1: Experimental results. The columns represent: (1) the system being verified, (2) # state variables, (3) # modules, (4) # covers of initial set, (5) # total simulation boxes, and (6) run time.

Systems	# V	# N	# C	# sim	RT (s)
Linear syn. I	16	4	128	45440	129.2
Linear syn. II	24	6	128	45649	135.1
Linear syn. delay I	16	4	64	23168	103.1
Linear syn. delay II	16	4	128	46304	205.8
Nonlinear cardiac delay I	6	3	16	6134	22.0
Nonlinear cardiac delay II	16	8	24	13563	42.5

Sample reach tubes computed for the linear synchronization example are shown in Figures 3.3 and 3.4. The blue trace (center) is the simulation of the actual system and the red lines give the upper- and lower-bounds computed from the reduced model. These are networks of two four-dimensional modules with linear dynamics $\dot{x}_1 = A_1x_1 + B_1u_1$ and $\dot{x}_2 = A_2x_2 + B_2u_2$. The modules are connected with an inter-modular gain $g > 0$ and a delay $d \geq 0$, such

that $u_1(t) = gx_2(t-d)$ and $u_2(t) = gx_1(t-d)$. For a network with Hurwitz A_1 and A_2, as seen in Figure 3.3, the diameter of the computed reach tube (for the same size of the initial set) grows quickly with the gain, but less so with delay. The computed reach tubes of networks with an unstable module are illustrated in Figure 3.4. From the construction of the reduced model (Definition 3.6), if a module is unstable with $\dot{\beta}_i(\delta, t) \geq 0$, the corresponding state component μ_i of the reduced model is unstable. Thus, even though the whole network may be stable, the reduced model will be unstable. In Figure 3.4, the reach tube in (a), which is corresponding to a smaller delay, is slightly thinner than the one in (b). From the trajectories (blue curves) in Figure 3.4, the whole network corresponding to (a) is stable however the reduced model does not capture this. This shows that our approach performs relatively well when either the individual modules are stable or the inter-modular gains are small.



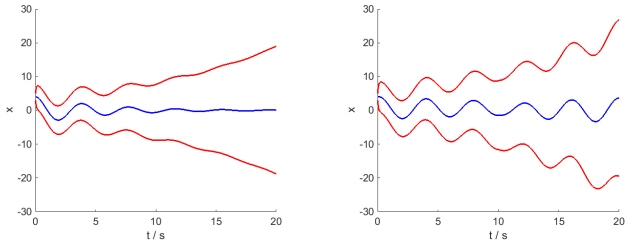
(a) Reach tube for network with delay $d = 0$ and gain $g = 1$

(b) Reach tube for network with delay $d = 0$ and gain $g = 0.5$

(c) Reach tube for network with delay $d = 10s$ and gain $g = 0.5$

Figure 3.3: Reach tubes for network with stable modules projected on one component of x_1. A_1 and A_2 are both Hurwitz. The blue curve is a trajectory, and the red curves outline a reach tube from a neighborhood.

Our implementation uses the built-in ODE solver of Matlab and these illustrative experiments assume that the error bounds of the computation in lines 5 and 20 are met by Matlab's ODE solver, but in reality this may not hold. However, from a sequence of sample simulation points, one can construct a sequence of simulation boxes satisfying the rigorous requirements of Definition 3.9 using methods such as the one presented in [80]. Alternatively, for one could use a validated ODE solver such as CAPD [72] as in the C2E2 verificaiton tool [52].



(a) Reach tube for network with delay 0

(b) Reach tube for network with delay $10s$

Figure 3.4: Reach tubes for network with an unstable module projected on one component of x_1. A_2 is Hurwitz while A_1 is not. The blue curve is a trajectory, and the red curves outline a reach tube from a neighborhood.

3.8 Summary

In this chapter, we presented a verification algorithm to prove bounded time invariance properties of (possibly unstable) nonlinear networked dynamical systems with delayed interconnection. Our method uses numerical simulations and IS discrepancy functions for the modules. IS discrepancy of a module $\mathcal{A}_i$, bounds the distance between two (possibly diverging) trajectories of $\mathcal{A}_i$ in terms of their initial states and inputs. It is closely related to the notion of input-to-state stability that is well studied in control theory, but an important distinction is that it does not require the subsystems or the overall system to be stable. Consequently, our construction of the lower-dimensional dynamical network $M(\delta)$ with the same delayed interconnection that gives a bound on the divergence of trajectories of $\mathcal{A}$, does not rely on any global properties like small-gain of the interconnection nor stability of $\mathcal{A}$, but instead only uses the individual IS discrepancy functions and the numerical simulations of $\mathcal{A}$ and $M(\delta)$. Further, we also show that by choosing appropriately small δ the over-approximations can be made arbitrarily precise; and therefore our verification algorithm is sound and relatively complete.

Chapter 4

PARTIAL ORDER REDUCTION-BASED INVARIANT VERIFICATION

In this chapter, we continue with invariant verification for networked cyber-physical systems (NCPS). In Chapter 3, we study a continuous model, namely networked dynamical systems, where the states evolve continuously over time. In contrast, here we focus on systems modeled by discrete time models or *labeled transition systems* (LTS). In this model, nodes (or processes) take actions to communicate with each other and to update the local and shared states of the system. In a asynchronous network, the nodes runs concurrently and the actions may interleave arbitrarily. Thus, the number of the possible action sequences can be extremely large. Similar to the invariant verification approach we presented in Chapter 3, we study the reach set computation problem for labeled transition systems. Computing the reach set for such systems is challenging mainly for two reasons. First, like the model of networked dynamical system (Definition 3.3), the set of initial states is uncountable and the number of executions (or trajectories) is uncountable. Hence explicit examination of every single execution is intractable. Second, the number of action sequences, even from a single initial state, is large owing to the combinatorial explosion arising from concurrency. In some cases, the number grows exponentially with the time bound. In this chapter, we propose a verification method to address both challenges which exploits the sensitivity of executions to the ordering of actions.

4.1 Enhance Partial Order Reduction with Metrics

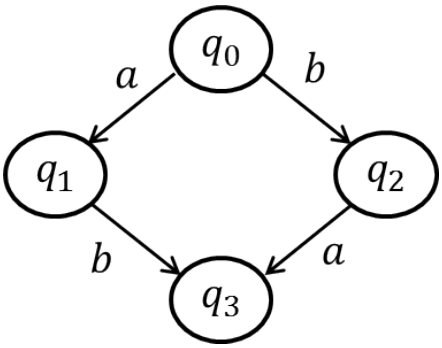
Consider an asynchronous network consisting of N components. In the case where each component takes an action independent of others, the entire network can take $N!$ different action sequences or executions. Checking the invariance property of the network then requires us to examine all these

possible action sequences. Partial order reduction methods were first introduced in the context of model checking concurrent softwares to reduce the number of action sequences needed to be explored in performing model checking [81, 34, 35]. It exploits the fact that some pairs of actions taken by different nodes may be commutative, and therefore the executions obtained by swapping them need not to be explored both times. Precisely, two actions a and b are said to be *independent* if from any state, the resulting state would be the same regardless of the order in which a and b are executed as illustrated in Figure 4.1a. This independence relation allows one to define relation over the set of executions which swaps independent actions. Equivalent executions from the same initial state reach the same final state. If we can show that the invariance properties are indifferent to equivalent executions, then checking invariance for a representative action sequence is sufficient for proving it for the whole equivalent class.

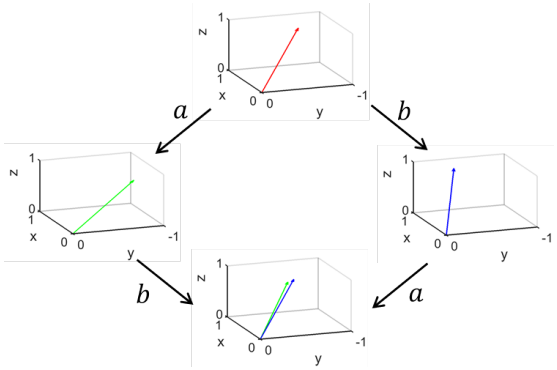
Partial order reduction has proven to be a powerful tool in software verification. It has been successfully applied to a variety of distributed systems, including leader election protocol [82], indexers [83], file systems [84], security protocol [85] and distributed schedulers [86]. In [84], a file transfer protocol with millions of states and transitions is verified in a matter of seconds using a partial order reduction method.

The conventional partial order methods have two limitations that hinder their application to cyber-physical systems. First, a pair of actions are considered independent only if they lead to *exactly* the same state regardless of the order of their execution. Such exactly independent action pairs are rare in CPS context, where the individual nodes often interact with a shared environment in the presence of noise and disturbances. This makes the partial order reduction methods ignore action pairs which lead to *nearly* but not exactly identical states. Second, there is no way to reason about executions with different initial states, even if they experience the same action sequences.

In this work, we develop partial order reduction techniques that take advantage of information about the sensitivity of the transitions of the individual agents in the system to prune executions that are approximately equivalent. First of all, we assume that there is a metric on the state space. We introduce the notion of *approximate independence* of actions. Specifically, with a parameter $\varepsilon \geq 0$ chosen by the user, two actions a and b are



(a) Resulting states are exactly the same for executing actions a and b in any order.



(b) The states are indicated by vectors. Action a rotate the state around x-axis for 20° and b rotate the state around z-axis for 20°. Executing actions a and b in different ordering result in states close to each other.

Figure 4.1: Exactly and approximately independent actions.

ε-independent if from any state, the resulting states are within ε distance regardless of the order in which actions a and b are executed. An example of approximate independent actions is illustrated in Figure 4.1b. Smaller of the parameter ε, the final states after executing ε-independent actions in any order are closer to each other. The conventional (exact) independence relation, as illustrated in Figure 4.1a, is a special case of ε-independence relation with $\varepsilon = 0$. With this extension, we can construct equivalent classes of action sequences by swapping approximately independent actions. Although executions that follow equivalent action sequences may not necessarily reach the same final state, the final states should be close to each other.

Together with this approximate independence relation, we exploit the continuity of the transition functions for approximating reachable states. Roughly, the continuity of an action a captures the notion that executing action a from two states that are close to each other, should result in states that remain close. Using the continuity property, we are able to inductively compute the distance between a pair of executions using the distance between their initial states. Combining continuity with ε-independence relation, though careful analysis which will be presented in Section 4.6 we can quantify the distance between executions starting from initial states that are close to each other and follow equivalent action sequences.

In a nutshell, partial order reduction is model checking that exploits rep-

resentatives from equivalent classes of executions [23, 87]. The reduction is more efficient if a larger size of equivalent class can be represented by a particular execution. Now with the notions of ε-independent actions and continuity, we expand the representation power of a single execution. We will show examples in Section 4.8 that for models where the conventional partial order reduction does not apply, our method can efficiently compute an over-approximation of states reached by a class of executions close to a given representative.

The rest of this chapter is organized as follows. First in Section 4.2 we discuss related works. In Section 4.3, we introduce a modeling framework for infinite state systems and a notion of continuity of actions. Then in Section 4.4, we introduce the notion of ε-independent actions and equivalent classes of action sequences. In Sections 4.5-4.7, we gradually develop an algorithm to compute over-approximated reach set of all equivalent executions using a reduced set of action sequences. This algorithm is applied to verify invariance properties of a linear transition system and a heater control system in Section 4.8.

4.2 Related Works

There are two main classes of partial order methods proposed in the last few decades: persistent set methods and sleep set methods. For each state of the system, the *persistent/ample set* methods compute a subset of enabled transitions, the persistent set (or ample set), such that the omitted transitions are independent to those selected [23, 82]. The reduced system which only considers the transitions in the persistent set is guaranteed to represent all behaviors of the original system. The persistent sets and the reduced systems are often derived by static analysis of the code. More recently, researchers developed the *sleep set* methods to avoid the static analysis [88, 89, 83]. These methods examine the history of actions taken by an execution and decide a set of actions that need to be explored in the future. The set of omitted actions is the sleep set. The persistent set and sleep set methods can be used complementary [83].

In the context of cyber-physical systems, robustness analysis are developed to quantify the closeness of executions with disturbances. The authors

of [90] presented an algorithm to compute the output deviation with bounded disturbance. The algorithm combines symbolic execution and optimization techniques. In [91], the authors introduce the robustness analysis to many classic algorithms and proposed a automated framework to prove robustness for softwares. Later in [92], this technique is extended to verify robustness of networked systems.

4.3 Infinite State Transition Systems

We start this chapter by introducing a *labeled transition system* as a mathematical model of concurrent NCPS. The state of the labeled transition system can be finite-valued or real-valued, that is, they are hybrid. The states evolve according to (possibly nondeterministic) actions and their corresponding transition functions. This framework can capture distributed consensus and flocking algorithms [93, 94, 95], clock synchronization protocols [96, 97], and other distributed control systems [98, 99]. We define a discrete version of discrepancy functions. Roughly, discrepancy functions we use here quantify the continuity property for the transition functions of the system.

4.3.1 Labeled Transition Systems

We will study a class of infinite state transition systems where each variable of the system either takes values in a finite set or is real-valued. The initial set and the set of executions can be infinite.

Definition 4.1. *A* Labeled Transition System $\mathcal{A}$ *is a tuple* $\langle X \cup L, \Theta, A, \rightarrow \rangle$ *where*

(i) X *is a set of* real-valued variables *and* L *is a set of* finite-valued variables. $Q = Val(X \cup L)$ *is the* set of states,

(ii) $\Theta \subseteq Q$ *is a compact set of* initial states,

(iii) A *is a finite set of* actions, *and*

(iv) $\rightarrow \subseteq Q \times A \times Q$ *is a* transition relation.

A state $q \in Q$ is a valuation of the real-valued and finite-valued variables. We denote $q.X \triangleq q \lceil X$ and $q.L \triangleq q \lceil L$ respectively, the continuous (real-valued) and discrete (finite-valued) part of the state. As we discussed in Chapter 2, the continuous part $q.X$ can be seen as a vector in $\mathbb{R}^{|X|}$ with some fixed ordering. For $\delta \geq 0$ and $q \in Q$, we define the *δ-neighborhood* of q as $\mathcal{B}_\delta(q) \triangleq \{q' \in Q : q'.L = q.L \wedge |q'.X - q.X| \leq \delta\}$. The norm $|\cdot|$ is the standard p-norm with arbitrary $p \in [1, \infty]$ chosen by the user. For any $(q, a, q') \in \rightarrow$, we write $q \xrightarrow{a} q'$. For any action $a \in A$ we define $guard(a) = \{q \in Q \mid \exists q' \in Q, q \xrightarrow{a} q'\}$ as the *guard* of a, which is the set of states where a can occur. An action a is *deterministic* if for any state $q \in Q$, at most one state $q' \in Q$ satisfies $q \xrightarrow{a} q'$. A deterministic action is specified by a guard and a *transition function*. In this chapter we will overload the name of an action a with its transition function. Thus for each $q \in guard(a)$, $q \xrightarrow{a} a(q)$. In this chapter, we will make the following assumptions.

Assumption 4.1. *All the actions in A are deterministic. For each $a \in A$ and any $q, q' \in Q$, the transition function $a : Q \to Q$ satisfies*

$$\text{if } q.L = q'.L, \text{ then } a(q).L = a(q').L.$$

It is straightforward to see that any deterministic transitions system with no real-valued variables ($X = \emptyset$) satisfies Assumption 4.1, where the transition function $a(q)$ is defined as q' if $q \xrightarrow{a} q'$ and otherwise as q.

Let A^* denote the set of finite *action sequences* (also called *traces*). For an action sequence $\tau \in A^*$, we denote by $len(\tau) \in \mathbb{N}$ the length of τ. For any for $i \in [len(\tau)]$, $\tau(i)$ is the i-th action in τ.

For a deterministic transition system, a state $q_0 \in Q$ and an action sequence $\tau = a_0 a_1 \ldots a_{n-1}$ uniquely specify a *potential execution* as an alternating sequence $\xi = q_0, a_0, q_1, a_1, \ldots, a_{n-1}, q_n$ where for each $i \in [n]$ $a_i(q_i) = q_{i+1}$. We denote such a potential execution by $\xi_{q_0,\tau}$. An *execution* is a potential execution such that (i) for $i \in [n]$, $q_i \in guard(a_i)$, and (ii) the first state q_0 is in the initial set Θ. For any potential execution ξ, $trace(\xi)$ denote its action sequence, that is $trace(\xi_{q_0,\tau}) = \tau \in A^*$. The length of ξ equals to the length of its trace. For $i \in [len(\xi)]$, $\xi(i) = q_i$ is the state visited by ξ after the i-th transition. The first and last state on ξ are denoted respectively by $\xi.\mathsf{fstate} = \xi(0)$ and $\xi.\mathsf{lstate} = \xi(len(\xi))$.

For a subset of initial states $S \subseteq \Theta$ and a time bound $T \geq 0$, we denote by

$\mathsf{Execs}(S,T)$ as the set of length T executions of the system from initial states S. We denote by $\mathsf{Traces}(S,T) \triangleq \{trace(\xi) \mid \xi \in \mathsf{Execs}(S,T)\}$ as the set of length T traces that are taken by executions from S. We denote the *reach set at time* T by $\mathsf{Reach}(S,T) \triangleq \{\xi.\mathsf{lstate} \mid \xi \in \mathsf{Execs}(S,T)\}$. In this chapter, we present an algorithm to over-approximate the reach set $\mathsf{Reach}(\Theta,T)$ using partial order reduction.

```
1  automata Linear(n ∈ ℕ, N ∈ ℕ)                transitions                          1
     variables                                    a_i for each i ∈ [N]
3      x : ℝ^n;                                     pre  d_i = false                 3
       d : 𝔹^N;                                     eff  x := A_i x ∧ d_i := true;
5    initially                                    a_⊥                              5
       x_i ∈ [−4, 4] for each i ∈ [n];              pre ∧_{i∈[N]} d_i:
7      d_i := false for each i ∈ [n];               eff d_i := false for each i ∈ [N]; 7
```

Figure 4.2: Transition system of consensus.

Example 4.1 (Asynchronous Linear Transition System). An asynchronous linear transition system is presented in Figure 4.2. The system two vectors of variables, a real-valued vector x with valuations in $\mathbb{R}^n$ and a Boolean-valued vector d. The set of action is $\{a_i\}_{i\in[N]} \cup \{a_\perp\}$. The guard and transition function of each action are defined by the precondition (**pre**) and effect (**eff**) statements. For each $i \in [N]$, the action a_i modifies x as $A_i x$, where A_i is an $n \times n$ matrix. The Boolean d_i's are used to ensure that all the actions a_i occur before any can be repeated. If for all $i \in [N]$ $d_i = true$, then the action $a_\perp$ is enabled and it resets d_i to *false*.

It can be checked that the transitions are deterministic. For any $q, q' \in Q$ with $q.d = q'.d$, we will show that Assumption 4.1 holds. For any $i \in [N]$, action a_i sets d_i to *true* and keeps all other components unchanged, hence $a_i(q).d = a_i(q').d$. Also, the action $a_\perp$ resets all variables d_i, hence $a_\perp(q).d = a_\perp(q').d$. Hence, this transition system satisfies Assumption 4.1.

For an instance of the system with $N = 3$, an example action sequence can be $\tau = a_0a_2a_1a_\perp a_1a_0a_2a_\perp$, where the reset action $a_\perp$ is applied exactly once after all of the other actions $\{a_0, a_1, a_2\}$ are applied once. Thus, the linear transformations can be applied in any order and one application of each constitutes a "round". This can be viewed as an abstraction of iterative consensus [100, 46], where N processes perform computations on a shared state. We note that the system will remain satisfying Assumption 4.1 if we replace A_i with a nonlinear transition function $a_i : \mathbb{R}^n \to \mathbb{R}^n$.

4.3.2 Discrepancy Functions

As we discussed in Chapter 3 and the authors show in [27, 52], for computing reach set from an uncountable set of initial states, one can exploit the sensitivity of the executions on initial states. In the context of labeled transition systems, discrepancy functions capture the distance between continuous states after an action is applied to a pair of states.

Definition 4.2. *For an action $a \in A$, a function $\beta_a : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a* discrepancy function *if for any pair of states $q, q' \in Q$ with $q.L = q'.L$,*

(i) $|a(q).X - a(q').X| \leq \beta_a(|q.X - q'.X|)$, *and*

(ii) $\beta_a \to 0$ *as* $|q.X - q'.X| \to 0$.

Roughly, discrepancy functions captures that executing action a from two states that are close to each other should result in states that remain close. With discrepancy functions, we can reason about distance between executions that share the same trace using their initial distance.

Proposition 4.1. *Fix any time bound $T \geq 0$ and any action sequences $\tau = a_0 a_1 a_2 \dots a_T$ equipped with discrepancy functions $\{\beta_{a_t}\}_{t=0}^{T}$. For any pair of states $q, q' \in Q$ with $q.L = q'.L$, the last states of the pair of potential executions $\xi = \xi_{q,\tau}$ and $\xi' = \xi_{q',\tau}$ satisfy*

$$\xi.\,\mathsf{lstate}\,.L = \xi'.\,\mathsf{lstate}\,.L, \text{ and}$$

$$|\xi.\,\mathsf{lstate}\,.X - \xi'.\,\mathsf{lstate}\,.X| \leq \beta_{a_T}\beta_{a_{T-1}} \dots \beta_{a_0}(|q.X - q'.X|).$$

The proposition described above can be proved by applying the property of discrepancy functions along the action sequence τ. This proposition allows us to compute the distance between a pair of potential executions from their initial distance. We will derive the discrepancy functions for the linear transition systems in the following example.

Example 4.2. Consider an instance of the asynchronous linear transition system presented in Example 4.1 with $n = 3$ and $N = 3$. We will use the

standard 2-norm on the state space $\mathbb{R}^3$. Let the matrices A_i be

$$A_0 = \begin{bmatrix} 0.2 & -0.2 & -0.3 \\ -0.2 & 0.2 & -0.1 \\ -0.3 & -0.1 & 0.3 \end{bmatrix}, A_1 = \begin{bmatrix} 0.2 & 0.3 & 0.2 \\ 0.3 & -0.2 & 0.3 \\ 0.2 & 0.3 & 0 \end{bmatrix}, A_2 = \begin{bmatrix} -0.1 & 0 & 0.4 \\ 0 & 0.4 & -0.2 \\ 0.4 & -0.2 & -0.1 \end{bmatrix}.$$

We note that all the three matrices are stable. For any $a_i \in A$ and any state $q, q' \in Q$ with $q.L = q'.L$,

$$|a_i(q).X - a_i(q').X|_2 \leq |A_i|_2 |q.X - q'.X|_2.$$

We note that the induced 2-norms of the matrices are $|A_0|_2 = 0.57, |A_1|_2 = 0.56, |A_2|_2 = 0.53$. For any $v \in \mathbb{R}_{\geq 0}$, the discrepancy functions of a_0, a_1, a_2 are defined as follows,

$$\beta_{a_0}(v) = 0.57v, \beta_{a_1}(v) = 0.56v, \text{ and } \beta_{a_2}(v) = 0.53v.$$

4.4 Independent Actions and Close Executions

Central to partial order methods is the notion of independent actions. In the conventional sense, a pair of actions are independent if the resulting state is exactly the same regardless of the order in which the actions are executed. In this section, we extend this notion with an approximation parameter $\varepsilon \geq 0$, such that a pair of actions are ε-independent if the resulting states are within ε distance after swapping the order of actions.

4.4.1 Approximately Independent Actions

Definition 4.3. *For $\varepsilon \geq 0$, two distinct actions $a, b \in A$ are* ε-independent*, denoted by $a \overset{\varepsilon}{\sim} b$, if for any state $q \in Q$,*

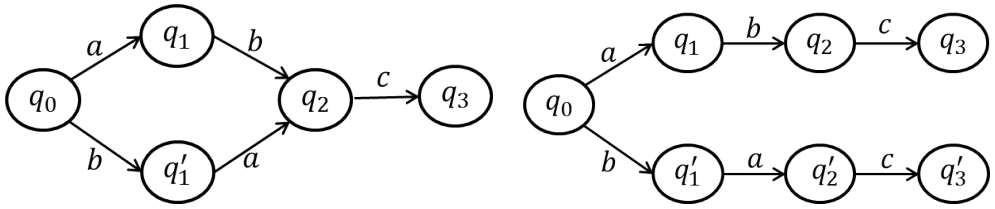
(i) (Commutativity) $ab(q).L = ba(q).L$, and

(ii) (Closeness) $|ab(q).X - ba(q).X| \leq \varepsilon$.

This notion of ε-independence has two extensions form the standard definition of independent actions (see e.g. Definition 8.3 of [24]). The main

extension is that the continuous states of $ab(q)$ and $ba(q)$ need not match up perfectly. Instead, for a parameter $\varepsilon \geq 0$, the continuous states are required to be within ε distance. If the two actions a, b are ε-approximately independent with $\varepsilon = 0$, Definition 4.3 becomes the commutative condition in the standard definition of independent actions.

We note that this extension introduces a side effect that swapping independent actions may disturb the enableness of the future actions. An example is illustrated in Figure 4.3. Suppose action a and b are ε-independent with $\varepsilon = 0$, as illustrated in Figure 4.3a. If $\xi_{q_0,abc}$ is an execution, then action c is enabled at state q_2. After swapping actions a and b the potential execution $\xi_{q_0,bac}$ still visits q_2 and action c remains to be enabled. However, if $\varepsilon > 0$, as illustrated in Figure 4.3b, swapping actions a and b makes the execution deviate from the state q_2 to q_2'. Then, whether action c would be enabled at q_2' is unknown. Hence, if $\xi_{q_0,abc}$ is an execution, that does not imply that the potential execution $\xi_{q,bac}$ is also an execution. Later in this chapter, we will compute a ball around an execution $\xi_{q_0,abc}$ such that the ball over-approximate all potential executions derived by swapping ε-independent action pairs of ξ. Due to this side effect, the tube may contains potential executions that need not to be included. Despite that, in Section 4.6, we are able to show that for small enough ε, this side effect can be made negligible and the reach set can be over-approximated precisely.



(a) If $a \overset{\varepsilon}{\sim} b$ with $\varepsilon = 0$, swapping actions a and b does not affect future actions.

(b) If $a \overset{\varepsilon}{\sim} b$ with $\varepsilon > 0$, swapping actions a and b may change the enableness condition of future actions.

Figure 4.3: The enableness condition guarantees that swapping independent actions in an execution results in a valid execution. However, the same property does not hold for approximately independent actions.

The second extension we made in Definition 4.3 is that action b is not required to be enabled at state $a(q)$. As a consequence, for a pair of ε-independent actions a and b, if $\xi_{q_0,ab}$ is an execution, $\xi_{q_0,ba}$ may not be one.

As we discussed in the previous paragraph, swapping ε-independent actions does not maintain the enableness property of an execution any way. Hence, making this extension does not introduce additional loss of precision.

In the notion of ε-independence, the parameter $\varepsilon \geq 0$ captures the degree of the approximation. For a smaller ε, the independent relation is more restrictive. For an action sequence $\tau \in A^*$ and an action $a \in A$, τ is ε-independent to a, written as $\tau \overset{\varepsilon}{\sim} a$, if τ is empty string or for every $i \in [len(\tau)]$, $\tau(i) \overset{\varepsilon}{\sim} a$.

It is clear that the approximate independence relation is symmetric. However, it need not to be transitive. We give instances of both transitive and non-transitive independence relations in the following example.

Example 4.3. We will discuss the approximately independent actions in Example 4.2. Fix any $i, j \in [N]$ such that $i \neq j$ and any state $q \in Q$. It can be checked that the following holds:

$$a_i a_j(q).d_k = a_j a_i(q).d_k = \begin{cases} true, & k \in \{i, j\} \\ q.d_k, & otherwise. \end{cases}$$

Hence, we have $a_i a_j(q).d = a_j a_i(q).d$ and the commutativity condition of Definition 4.3 holds for any pair of actions a_i and a_j.
We will examine the closeness condition with an instance of the system. Notice that for any $q \in Q$, switching the order of a_i, a_j leads to

$$|a_i a_j(q).x - a_j a_i(q).x|_2 = |(A_i A_j - A_j A_i)q.x|_2 \leq |A_i A_j - A_j A_i|_2 |q.x|_2. \quad (4.1)$$

If the matrices A_i and A_j commute, then a_i and a_j are ε-approximately independent with $\varepsilon = 0$. If Q is a compact set or the system has a bounded invariant set, then $|q.X|_2$ is bounded and there always exists a finite $\varepsilon \geq 0$ such that $a_i \overset{\varepsilon}{\sim} a_j$. Consider the matrices A_0, A_1, A_2 presented in Example 4.2. We note that A_0, A_1, A_2 are all stable matrices. Fix any state q and any $i \in [3]$. The change in the squared 2-norm of the states after the transition is

$$|a_i(q).x|_2^2 - |q.x|_2^2 = q.x^\top A_i^\top A_i q.x - q.x^\top q.x = q.x^\top (A_i^\top A_i - I_3) q.x.$$

It can be checked that, for each $i \in [3]$, $A_i^\top A_i - I_3$ is a negative definite

matrix. Hence $|a_i(q).x|_2 \leq |q.x|_2$ holds for each $i \in [3]$ and any state q. That is, the norm of the continuous state is non-increasing. Suppose initially $x \in [-4, 4]^3$ then the 2-norm of the initial state is bounded by the value $4\sqrt{3}$. Since the norm of state is non-increasing in any transitions, we can show that $Inv = \{x \in \mathbb{R}^3 : |x|_2 \leq 4\sqrt{3}\}$ is an invariant of the system. Thus, from Equation (4.1), we have $|a_0a_1(q).x - a_1a_0(q).x|_2 \leq 0.1$, $|a_0a_2(q).x - a_2a_0(q).x|_2 \leq 0.07$, and $|a_1a_2(q).x - a_2a_1(q).x|_2 \leq 0.17$. Thus, with $\varepsilon = 0.1$, it follows that $a_0 \overset{\varepsilon}{\sim} a_2$ and $a_1 \overset{\varepsilon}{\sim} a_2$. Here the $\overset{\varepsilon}{\sim}$ relation is not transitive. On the other hand, if we choose $\varepsilon = 0.2$, then any pair of actions in a_0, a_1, a_2 is ε-independent, where $\overset{\varepsilon}{\sim}$ is transitive.

Definition 4.3 implies that from any state q, executing two ε-independent actions in either order, we end up in states that are within ε distance. In the following proposition, we examine potential executions with different initial states.

Proposition 4.2. *For a pair of ε-independent actions $a, b \in A$, two states $q, q' \in Q$ with $q.L = q'.L$, we have*

(i) $ba(q).L = ab(q').L$, *and*

(ii) $|ba(q).X - ab(q').X| \leq \beta_b\beta_a(|q.X - q'.X|) + \varepsilon$,

where β_a, β_b are discrepancy functions of a, b respectively.

Proof. Fix any pair of states $q, q' \in Q$ such that $q.L = q'.L$. Since $a \overset{\varepsilon}{\sim} b$, we have $ba(q).L = ab(q).L$. Using Assumption 4.1 twice, we have $ab(q).L = ab(q').L$.
Using triangular inequality, we have $|ba(q).X - ab(q').X| \leq |ba(q).X - ba(q').X| + |ba(q').X - ab(q').X|$. The first term is bounded by $\beta_b\beta_a(|q.X - q'.X|)$ using the property of discrepancy functions. The second term is bounded by ε by definition of approximate independent actions. Therefore $|ba(q).X - ab(q').X| \leq \beta_b\beta_a(|q.X - q'.X|) + \varepsilon$. □

4.4.2 Equivalent Action Sequences and Close Executions

For a set of ε-approximately independent actions, we define an equivalence relation on the space of finite action sequences A^*.

Definition 4.4. *For any $\varepsilon \geq 0$, we define a relation $R \subseteq A^* \times A^*$ such that $\tau R \tau'$ iff there exists $\sigma, \eta \in A^*$ and $a, b \in A$ such that*

$$a \overset{\varepsilon}{\sim} b,\ \tau = \sigma ab\eta,\ \text{and } \tau' = \sigma ba\eta.$$

We define an equivalence relation on traces $\overset{\varepsilon}{\equiv} \subseteq A^* \times A^*$ called ε-equivalence*, as the reflexive and transitive closure of R.*

That is, a pair of traces $\tau, \tau' \in A^*$ is ε-equivalent if we can construct τ' from τ by swapping consecutive ε-independent actions. In the following proposition, we show that two potential executions with the same initial discrete state (location) and equivalent trace share the same final locations.

Proposition 4.3. *Fix any potential executions $\xi = \xi_{q_0,\tau}$ and $\xi' = \xi_{q_0',\tau'}$. If ξ and ξ' have the same initial location and equivalent traces, then they have the same final location. Precisely,*

$$\text{if } q_0.L = q_0'.L \text{ and } \tau \overset{\varepsilon}{\equiv} \tau',\ \text{then } \xi.\mathsf{lstate}.L = \xi'.\mathsf{lstate}.L.$$

Proof. If $\tau = \tau'$, then the proposition directly follows Assumption 4.1. Suppose $\tau \neq \tau'$, from Definition 4.4, there exists a sequence of action sequences $\tau_0, \tau_1, \ldots, \tau_k$ to join τ and τ' by swapping neighboring approximately independent actions. Precisely the sequence $\{\tau_i\}_{i=0}^k$ satisfies

(i) $\tau_0 = \tau$ and $\tau_k = \tau'$, and

(ii) for each pair τ_i and τ_{i+1}, there exists $\sigma, \eta \in A^*$ and $a, b \in A$ such that $a \overset{\varepsilon}{\sim} b$, $\tau_i = \sigma ab\eta$, and $\tau_{i+1} = \sigma ba\eta$.

From Definition 4.3, swapping approximately independent actions maintains the location in the final state. Hence for any $i \in [k]$, $\xi_{q_0,\tau_i}.\mathsf{lstate}.L = \xi_{q_0,\tau_{i+1}}.\mathsf{lstate}.L$. Therefore, $\xi.\mathsf{lstate}.L = \xi'.\mathsf{lstate}.L$. □

Informally, a pair of potential executions that take equivalent action sequences should be close to each other. The following definition captures these pairs of potential executions.

Definition 4.5. *For $\delta, \varepsilon \geq 0$, a pair of potential executions $\xi = \xi_{q_0,\tau}$ and $\xi' = \xi_{q_0',\tau}$ are* (δ, ε)-close *to each other, denoted by $\xi \overset{\delta,\varepsilon}{\approx} \xi'$, if*

$$q_0.L = q_0'.L,\ |q_0.X - q_0'.X| \leq \delta,\ \text{and } \tau \overset{\varepsilon}{\equiv} \tau'.$$

That is, two potential executions ξ, ξ' are (δ, ε)-close if their first states lie in the δ-neighborhood of each other, and their traces are ε-equivalent. It follows from Proposition 4.3 that the final locations of any pair of (δ, ε)-close potential executions are the same. In Section 4.6, we quantify the distance between (δ, ε)-close potential executions. Then, by bloating a single potential execution, we can over-approximate the reach set of all potential executions close to it.

Example 4.4. In Example 4.3, we show that $a_0 \overset{\varepsilon}{\sim} a_2$ and $a_1 \overset{\varepsilon}{\sim} a_2$ with $\varepsilon = 0.1$. We consider the following two executions of the system:

$$\xi = q_0, a_2, q_1, a_1, q_2, a_0, q_3, a_\perp, q_4, \text{ and } \xi' = q'_0, a_1, q'_1, a_0, q'_2, a_2, q'_3, a_\perp, q'_4.$$

That is, the traces of the executions are $\mathit{trace}(\xi) = a_2 a_1 a_0 a_\perp$ and $\mathit{trace}(\xi') = a_1 a_0 a_2 a_\perp$. For $\varepsilon = 0.1$, we have $a_2 a_1 a_0 a_\perp \overset{\varepsilon}{\equiv} a_1 a_2 a_0 a_\perp$ and $a_1 a_2 a_0 a_\perp \overset{\varepsilon}{\equiv} a_1 a_0 a_2 a_\perp$. Since the equivalence relation $\overset{\varepsilon}{\equiv}$ is transitive, we have $\mathit{trace}(\xi) \overset{\varepsilon}{\equiv} \mathit{trace}(\xi')$. Suppose $q_0 \in \mathcal{B}_\delta(q'_0)$, then ξ and ξ' are (δ, ε)-close executions with $\varepsilon = 0.1$.

4.5 Interleaving Independent Actions

In this section, we present several results that help partial order reduction in model checking labeled transition systems. We will study the distance between potential executions which are derived by inserting a single action to the same execution at different positions. Building up on this result, for any $\delta, \varepsilon \geq 0$ and an execution ξ, we develop an inductive method for computing distance between ξ and its (δ, ε)-close executions. Throughout this section, we study a simplified case where the actions are mutually ε-independent. Later in Section 4.6, we generalize the this result to actions defined with an arbitrary ε-independence relation.

We start with some definitions related to a set of discrepancy functions. For a finite set of discrepancy functions $\{\beta_a\}_{a \in S}$ corresponding to a set of actions $S \subseteq A$, we define $\beta = \max_{a \in S}\{\beta_a\}$ as the upper bound of discrepancy functions. For an $n \geq 0$ and a function β defined as above, we define a function $\gamma_n = \sum_{i=0}^{n} \beta^i$. Recall from Chapter 2 that β^i is the nested form of β such that $\beta^i = \beta\beta^{i-1}$ for $i \geq 1$ and β^0 is the identity mapping. We

note that for any $n \in \mathbb{N}$, the function γ_n is uniquely specified by a set of discrepancy functions $\{\beta_a\}_{a\in S}$. Using the properties of discrepancy functions as in Definition 4.2, we can show the following properties of $\{\gamma_n\}_{n\in\mathbb{N}}$.

Proposition 4.4. *Fix any finite subset of discrepancy functions $\{\beta_a\}_{a\in S}$ with $S \subseteq A$. Let $\beta = \max_{a\in S}\{\beta_a\}$ be the maximum function. For any $n \geq 0$, $\gamma_n = \sum_{i=0}^{n} \beta^i$ satisfies*

(i) for any $\varepsilon \in \mathbb{R}_{\geq 0}$ and any $n \geq n' \geq 0$, $\gamma_n(\varepsilon) \geq \gamma_{n'}(\varepsilon)$, and

(ii) $\lim_{\varepsilon\to 0} \gamma_n(\varepsilon) = 0$.

Proof. (i) For any $n \geq 1$, we have $\gamma_n - \gamma_{n-1} = \beta^n$. Since $\beta^n = \max_{a\in S}\{\beta_a\}$ for some finite S, using Definition 4.2, β^n takes only non-negative values. Hence, the sequence of functions $\{\gamma_n\}_{n\in\mathbb{R}_{\geq 0}}$ is non-decreasing.
(ii) Using the property of discrepancy functions, we have $\lim_{\varepsilon\to 0} \beta(\varepsilon) = 0$. By induction on the nested functions, we have $\lim_{\varepsilon\to 0} \beta^i(0)$ for any $i \geq 0$. Hence for any $n \in \mathbb{R}_{\geq 0}$, $\lim_{\varepsilon\to 0} \gamma_n(\varepsilon) = \lim_{\varepsilon\to 0} \sum_{i=0}^{n-1} \beta^i(\varepsilon) = 0$. □

4.5.1 Insertion of Independent Action

We will quantify the distance of two potential executions with equivalent traces using the function γ_n defined above. Our first step involves computing the distance between two potential executions after inserting a single action at different position. We study a simplified case where the inserting action is independent to all other actions in the execution.

Lemma 4.5. *Fix any $\varepsilon \geq 0$, any initial state $q_0 \in Q$, any action $a \in A$ and any action sequence $\tau \in A^*$ with length $n \geq 1$. If a and τ are ε-independent, then the potential executions $\xi = \xi_{q_0,\tau a}$ and $\xi' = \xi_{q_0,a\tau}$ satisfy*

(i) $\xi'.\mathsf{lstate}.L = \xi.\mathsf{lstate}.L$, and

(ii) $|\xi'.\mathsf{lstate}.X - \xi.\mathsf{lstate}.X| \leq \gamma_{n-1}(\varepsilon)$, where γ_{n-1} corresponds to the set of discrepancy functions $\{\beta_c\}_{c\in\tau}$ for the actions in τ.

Proof. Part (i) directly follows from Proposition 4.3. We will prove part (ii) by induction on the length of the action sequence τ.
Base: For any action sequence τ of length 1, ξ and ξ' are of the form $\xi = q_0, b_0, q_1, a, q_2$ and $\xi' = q_0, a, q_1', b_0, q_2'$. Since $a \overset{\varepsilon}{\sim} b_0$, it follows from

Definition 4.3 that $|q_2'.X - q_2.X| \leq \varepsilon$. Recall from the preliminary that $\gamma_0(\varepsilon) = \beta^0(\varepsilon) = \varepsilon$. Hence $|q_2'.X - q_2.X| \leq \gamma_0(\varepsilon)$ holds for action sequence τ with length $n = 1$.

Induction: Suppose the lemma holds for any τ with length at most $n-1$. Fixed any $\tau = b_0 b_1 \dots b_{n-1}$ of length n, we will show the lemma holds for τ. Let the potential executions $\xi = \xi_{q_0,\tau a}$ and $\xi' = \xi_{q_0,a\tau}$ be of the form

$$\xi = q_0, b_0, q_1, b_1, ..., b_{n-1}, q_n, a, q_{n+1},$$
$$\xi' = q_0, a, q_1', b_0, q_2', b_1, ..., b_{n-1}, q_{n+1}'.$$

It suffice to prove that $|\xi.\mathsf{lstate}.X - \xi'.\mathsf{lstate}.X| = |q_{n+1}.X - q_{n+1}'.X| \leq \gamma_{n-1}(\varepsilon)$. We first construct a potential execution $\xi'' = \xi_{q_0,b_0 a b_1 \dots b_{n-1}}$ by swapping the first two actions of ξ'. Then, ξ'' is of the form

$$\xi'' = q_0, b_0, q_1, a, q_2'', b_1, ..., b_{n-1}, q_{n+1}''.$$

The potential executions ξ, ξ' and ξ'' are illustrated in Figure 4.4.

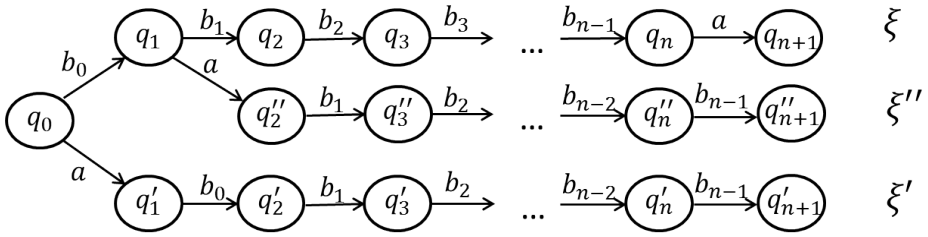


Figure 4.4: Executions $\xi = \xi_{q_0,\tau a}, \xi' = \xi_{q_0,a\tau}$, and ξ'' which is constructed by swapping the first two actions in ξ'.

We first compare the potential executions ξ and ξ''. Notice that, ξ and ξ'' share a common prefix q_0, b_0, q_1. Starting from q_1, the action sequence of ξ'' is derived from $trace(\xi)$ by inserting action a in front of the action sequence $\tau' = b_1 b_2 \dots b_{n-1}$. Since $\tau' \overset{\varepsilon}{\sim} a$, applying the induction hypothesis on the length $n-1$ action sequence τ', we get

$$|q_{n+1}.X - q_{n+1}''.X| \leq \gamma_{n-2}(\varepsilon). \tag{4.2}$$

Then, we compare the potential executions ξ' and ξ''. Since $b_0 \overset{\varepsilon}{\sim} a$, by applying the property of Definition 4.3 to the first two actions of ξ' and ξ'',

we have $|q_2'.X - q_2''.X| \leq \varepsilon$. We note that ξ' and ξ'' have the same suffix of action sequence from q_2' and q_2''. Using Proposition 4.1 from states q_2' and q_2'', we have

$$|q_{n+1}'.X - q_{n+1}''.X| \leq \beta_{b_1}\beta_{b_2}\ldots\beta_{b_n-1}(|q_2'.X - q_2''.X|) \leq \beta^{n-1}(\varepsilon). \tag{4.3}$$

Combining Equations (4.2) and (4.3) with triangular inequality, we have

$$\begin{aligned} |q_{n+1}.X - q_{n+1}'.X| &\leq |q_{n+1}.X - q_{n+1}''.X| + |q_{n+1}'.X - q_{n+1}''.X| \\ &\leq \gamma_{n-2}(\varepsilon) + \beta^{n-1}(\varepsilon) = \gamma_{n-1}(\varepsilon). \end{aligned} \tag{4.4}$$

Therefore, the lemma holds for the action sequence τ with length n, which completes the induction. □

In Lemma 4.6, we analyzed distance between executions after inserting one single action at different position, where the difference in the inserting positions is n. We show that the distance between these executions increases with n.

4.5.2 Permutation of Independent Actions

Using the results we presented in Section 4.5.1, we will compute an upper bound of the distance between (δ, ε)-close potential executions. We start with a formal definition of the quantity we are going to compute.

Definition 4.6. *For any $\delta, \varepsilon, r \in \mathbb{R}_{\geq 0}$ and any potential execution $\xi = \xi_{q_0,\tau}$, ξ is a* (δ, ε, r)-representative potential execution *if any potential execution ξ' that is (δ, ε)-close to ξ satisfies*

$$\xi'.\mathsf{lstate} \in \mathcal{B}_r(\xi.\mathsf{lstate}).$$

That is, for a (δ, ε, r)-representative potential execution ξ, the r-neighborhood of its last state $\mathcal{B}_r(\xi.\mathsf{lstate})$ contains the last states of all its (δ, ε)-close potential executions. The parameter r is the *representative radius* of the potential execution ξ. In the rest of this chapter, we will present algorithms to compute the representative radius and reach set.

The following lemma states a inductive way of constructing representative potential executions. Roughly, it shows how the representative radius r changes if an action a is appended to a potential execution ξ. We start with a special case where the appended action a is ε-independent of all other actions on ξ.

Lemma 4.6. *Fix any $\delta, \varepsilon, r \in \mathbb{R}_{\geq 0}$, any (δ, ε, r)-representative potential execution $\xi_{q_0,\tau}$, and any $a \in A$ such that $\tau \overset{\varepsilon}{\sim} a$. Then, $\xi = \xi_{q_0,\tau a}$ is a $(\delta, \varepsilon, r')$-representative potential execution with*

$$r' = \beta_a(r) + \gamma_{len(\tau)-1}(\varepsilon). \tag{4.5}$$

Proof. Fix any ξ' such that $\xi' \overset{\delta,\varepsilon}{\approx} \xi$ with initial state $q'_0 \in \mathcal{B}_\delta(q_0)$. It follows from Proposition 4.3 that $\xi'.\mathsf{lstate}.L = \xi.\mathsf{lstate}.L$. It suffice to prove that $|\xi'.\mathsf{lstate}.X - \xi.\mathsf{lstate}.X| \leq r'$.

Since $trace(\xi') \overset{\varepsilon}{\equiv} \tau a$, $trace(\xi')$ is in a form $\phi a \eta$ with some $\phi\eta \overset{\varepsilon}{\equiv} \tau$. We construct a potential execution $\xi'' = \xi_{q'_0,\phi\eta a}$. The three potential executions are illustrated in Figure 4.5.

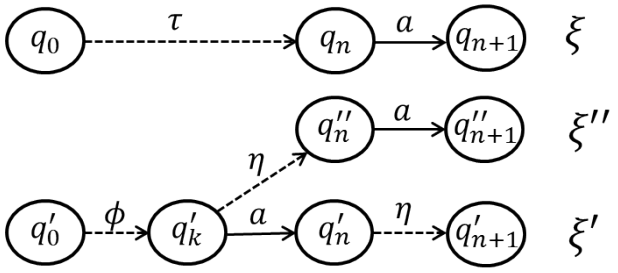


Figure 4.5: Execution ξ, its ε-equivalent execution ξ', and execution ξ'' that is constructed by swapping action a to the back of ξ'.

We note that the prefix (q_0, τ, q_n) of ξ is a (δ, ε, r)-representative potential execution. Since $\phi\eta \overset{\varepsilon}{\equiv} \tau$ and $q'_0 \in \mathcal{B}_\delta(q_0)$, it follows from Definition 4.6 that $|q_n.X - q''_n.X| \leq r$. Hence

$$|\xi.\mathsf{lstate}.X - \xi''.\mathsf{lstate}.X| \leq \beta_a(|q_n.X - q''_n.X|) \leq \beta_a(r). \tag{4.6}$$

On the other hand, we observe that the traces of ξ' and ξ'' differ only in the position of action a. Application of Lemma 4.5 on ξ' and ξ'' yields

$$|\xi'.\mathsf{lstate}.X - \xi''.\mathsf{lstate}.X| \leq \gamma_{len(\eta)-1}(\varepsilon) \leq \gamma_{len(\tau)-1}(\varepsilon). \tag{4.7}$$

Combining Equations (4.6) and (4.7) with triangular inequality, we have

$$|\xi.\,\mathsf{lstate}\,.X - \xi'.\,\mathsf{lstate}\,.X| \leq \beta_a(r) + \gamma_{len(\tau)-1}(\varepsilon).$$

□

In Lemma 4.6, we analyze how the representative radius changes after appending a single action a to a potential execution $\xi_{q_0,\tau}$. Here we consider the spacial case where action a is ε-independent to the entire trace τ. In the next section, we generalize this results to any action a and trace τ, without specific requirements on their independence condition.

4.6 Generalization of Executions

In this section, for any parameters $\delta_0, \varepsilon \geq 0$ and any potential execution ξ of length T, we will compute the representative radius δ_T for ξ, such that ξ is a $(\delta_0, \varepsilon, \delta_T)$-representative potential execution. Our method involves computing a sequence of representative radii $\{\delta_t\}_{t=0}^{T}$ inductively such that δ_t is the representative radius of the length t prefix of ξ.

Let action a be the t-th action on ξ and τ be the length t prefix of $trace(\xi)$. If action a is ε-independent to τ, then the representative radius δ_t can be computed from δ_{t-1} using Lemma 4.6. In this section, we generalize this result to the case where action a is not necessarily ε-independent to the whole sequence τ. First, we will introduce the notion of *anchor position* of a in τ, which is the left most position of action a in all equivalent traces of τ.

4.6.1 Anchor Position

In the proof of Lemma 4.6, we use the fact that any ε-equivalent trace of τa must be in a form of $\phi a \eta$. We observe that, the representative radius of $\xi_{q_0,\tau a}$ depends on the length of ϕ and η, as presented in Equation (4.7). For computing the representative radius, we need the maximum length of η, or equivalently the minimum length of ϕ. We introduce the notion of *anchor position* to capture this quantity.

For any sequence $\tau \in A^*$ and an action $a \in \tau$ in τ, we define $\tau.lastPos(a)$ as the largest index k such that $\tau(k) = a$. The *anchor position* is the first

possible position of a in any τ' that is equivalent to τ. We formally define this quantity as follows.

Definition 4.7. *For any action sequence $\tau \in A^*$ and any action $a \in A$, the* anchor position *of a on τ is*

$$\min_{\tau' \overset{\varepsilon}{\equiv} \tau a} \tau'.lastPos(a).$$

For any trace τa, its ε-equivalent traces can be derived by swapping consecutive ε-independent action pairs. Hence, the anchor position of a is the left most position it can be swapped to. Since any equivalent trace of τa can be written in a form $\phi a \eta$ with some $\phi, \eta \in A^*$, an equivalent way of defining the anchor position is as as following:

$$\min_{\phi a \eta \overset{\varepsilon}{\equiv} \tau a,\ a \notin \eta} len(\phi).$$

In Algorithm 4.1, we find the anchor position of action a on an action sequence τ. For any trace τ and action a, $anchor(\tau, a)$ constructs a trace $\phi \in A^*$. Initially ϕ is set to be the empty sequence. Iteratively, from the end of τ, we add action $\tau(t)$ to ϕ if it is not independent to the entire trace ϕa. We will prove that, length of ϕ gives the anchor position of action a on trace τ. The time complexity of the algorithm is at most $O(n^2)$, where n is the length of trace τ.

Algorithm 4.1 anchor(τ, a)

```
1: φ ← ⟨⟩;
2: //let T be the length of τ
3: for t = T − 1 : 0 do
4:     if ∃b ∈ φa, τ(t) ⊀^ε b then
5:         φ ← τ(t)φ;
6:     end if
7: end for
8: return len(φ);
```

Lemma 4.7. *For any action $a \in A$ and trace $\tau \in A^*$, the function $anchor(\tau, a)$ computes the anchor position of a on τ.*

Proof. For a trace τ and an action a, algorithm $anchor(\tau, a)$ constructs a

trace ϕ and returns its length. To prove that $len(\phi)$ gives the anchor position (p) of a on τ, we show both $len(\phi) \geq p$ and $len(\phi) \leq p$.

***len*(ϕ) $\geq$ *p*.** It suffice to prove the statement by constructing a trace η such that $\phi a\eta \overset{\varepsilon}{\equiv} \tau a$ and $a \notin \eta$. Let $\eta = \tau \backslash \phi$ be the remaining subsequence of τ after removing the actions in ϕ. We note that the ordering of actions in η is the same as that in τ. For each action $c \in \eta$, line 5 is not executed. Hence, for all actions $b \in \phi a$ which is originally to the right of c, we have $b \overset{\varepsilon}{\sim} c$. Therefore, action c can be swapped repeatedly to the right of action a. Repeat this process for all actions in η, we derive trace $\phi a\eta$ from the original trace τa. Therefore $\phi a\eta \overset{\varepsilon}{\equiv} \tau a$. In addition, we note that from Definition 4.3, an ε-independent action pair consists of two distinctive actions, which implies $a \overset{\varepsilon}{\not\sim} a$. Hence, for each occurrence $a \in \tau$, line 5 is not executed, that is, $a \notin \eta$. Therefore, the statement holds.

***len*(ϕ) $\leq$ *p*.** First, we convert any trace τa to a trace consists of only distinctive actions. If otherwise some action $b \in \tau a$ occurs more than once, we replace the occurrences as distinctive pseudo-actions $b_0, b_1, \ldots$, such that each b_i inherit the same independence relation from b and any pair of these pseudo-actions is not independent. In this way, we map an arbitrary trace τa to a trace consists of only distinctive actions. It can be checked that this mapping is bijective. Without loss of generality, we assume that the actions in τa are distinctive.

We prove $len(\phi) \leq p$ by contradiction. Suppose $len(\phi) > p$, then there exist traces ϕ', η' such that (i) $a \notin \eta'$, (ii) $\phi' a\eta' \overset{\varepsilon}{\equiv} \tau a$, and (iii) $len(\phi') < len(\phi)$. From (iii), there exists an action $c \in \phi \backslash \phi'$. If there are multiple choices of such actions, we choose the rightmost action c in ϕ. From lines 4 and 5, action c is in ϕ iff there exists another action $b \in \phi$ to the right of c such that $c \overset{\varepsilon}{\not\sim} b$. Since we choose action c as the rightmost action in ϕ that is not in ϕ', we have $b \in \phi'$. Originally in trace τa, action b is to the right of action c. As actions b and c are not ε-independent, in any equivalent trace $\phi' a\eta' \overset{\varepsilon}{\equiv} \tau a$, the relative position of them should not be changed. Hence in trace $\phi' a\eta'$, action b is also to the right of action c. However, since $b \in \phi'$ and $c \notin \phi'$, we have action c is to the right of action b in trace $\phi' a\eta'$. We derive a contradiction. Therefore, if the actions in τa are distinctive, $len(\phi) \leq p$.

□

Example 4.5. In Example 4.3, we show that $a_0 \overset{\varepsilon}{\sim} a_2$ and $a_1 \overset{\varepsilon}{\sim} a_2$ with

$\varepsilon = 0.1$. It can also be checked that $a_\perp$ is not ε-independent to any actions. Consider the anchor position of a_0 on the trace $\tau = a_\perp a_2 a_1$. We can swap a_0 ahead following the sequence $\tau a_0 = a_\perp a_2 a_1 a_0 \overset{\varepsilon}{\equiv} a_\perp a_1 a_2 a_0 \overset{\varepsilon}{\equiv} a_\perp a_1 a_0 a_2$. Since neither of $a_\perp$ and a_1 is independent to action a_0, it cannot be swapped any further ahead. The anchor position of a_0 is 2, corresponding to its position on $a_\perp a_1 a_0 a_2$. Algorithm 4.1 construct a trace $\phi = a_\perp a_1$, where the length of ϕ gives the correct anchor position 2.

4.6.2 Generalization of an Individual Execution

In this section, we present an algorithm to compute a ball centered at the final state of a potential execution ξ, such that the ball contains the final states of all (δ_0, ε)-close executions of ξ. Algorithm 4.2 takes inputs of an execution $\xi = \xi_{q_0,\tau}$, two parameters $\delta_0, \varepsilon \geq 0$, and a set of discrepancy functions $\{\beta_a\}_{a \in A}$. For each $t \in [len(\xi)]$, let a be the t-th action of ξ and τ be the prefix before a. The algorithm first finds a lower bound of the anchor position of a on τ, which is the leftmost position that action a can reach by swapping consecutive ε-independent action on τ. Using the anchor positions, we can compute a sequence of representative radii $\{\delta_t\}_{t=1}^T$ inductively. Our computation guarantees that for each $t \in \{1, 2, \ldots, T\}$, the length t prefixes of ξ is a $(\delta_0, \varepsilon, \delta_t)$-representative potential execution.

Algorithm 4.2 Generalization$(\xi, \delta_0, \varepsilon, \{\beta_a\}_{a \in A})$

```
1: β ← max{β_a};
2: //let T be the length ξ
3: for t ∈ [T] do
4:     //let the length (t + 1) prefix of trace(ξ) be τa
5:     k ← anchor(τ, a);
6:     if k = t then
7:         δ_{t+1} ← β_a(δ_t);
8:     else
9:         δ_{t+1} ← β_a(δ_t) + γ_{t−k−1}(ε);
10:    end if
11: end for
12: return B_{δ_T}(ξ(T));
```

We will establish the correctness of Algorithm 4.2. First, we will show that the sequence of $\{\delta_t\}_{t=1}^T$ is valid representative radii for the prefixes of ξ.

Lemma 4.8. *For any $t \leq len(\xi)$, the length t prefix of ξ is a $(\delta_0, \varepsilon, \delta_t)$-representative execution.*

Proof. For any $t \leq len(\xi)$, We prove the lemma by induction on t.
Base: For $t = 0$, the length 0 prefix of ξ is a single state q_0. Any ξ' (δ_0, ε)-close to q_0 is also a single state q'_0 with $q'_0 \in \mathcal{B}_{\delta_0}(q_0)$. Hence the lemma holds for $t = 0$.
Induction: Suppose the lemma holds for any prefix of ξ with length at most $t < len(\xi)$. We will prove the lemma holds for the length $t + 1$ prefix of ξ. Let q_0 be the initial state of ξ, τ be the length t prefix of $trace(\xi)$, and $a \in A$ is the $(t+1)$th action. Then $\xi_{t+1} = \xi_{q_0,\tau a}$ is the length $t+1$ prefix of ξ. The execution ξ_{t+1} is illustrated in Figure **??**. Fix any ξ' that is ξ' is (δ_0, ε)-close to ξ_{t+1}. From Proposition 4.3, $\xi'.\mathsf{lstate}.L = \xi_{t+1}.\mathsf{lstate}.L$. It suffice to prove that $|\xi'.\mathsf{lstate}.X - \xi_{t+1}.\mathsf{lstate}.X| \leq \delta_{t+1}$.

Since $trace(\xi') \overset{\varepsilon}{\equiv} \tau a$, action a is in the sequence $trace(\xi')$. Partitioning $trace(\xi')$ on the last occurrence of a, we get $trace(\xi') = \phi a \eta$ for some $\phi, \eta \in A^*$ with $a \notin \eta$. Since k is a lower bound of the anchor position, from Definition 4.7, the position of the last occurrence of a on $trace(\xi')$ is at least k. Hence we have $len(\phi) \geq k$ and $len(\eta) = t - len(\phi) \leq t - k$. We construct another potential execution $\xi'' = \xi_{\xi'.\mathsf{fstate},\phi\eta a}$. The executions ξ, ξ' and ξ'' are illustrated in Figure 4.6.

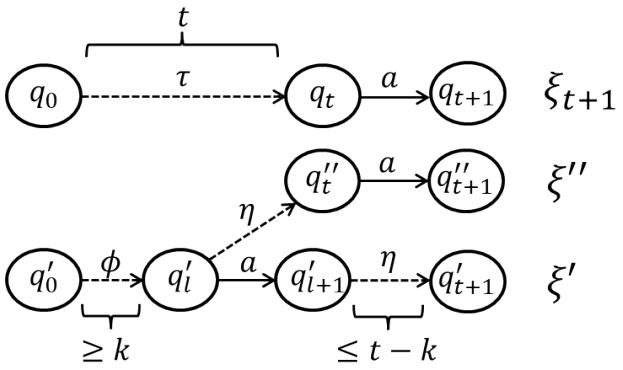


Figure 4.6: Illustration of the potential executions.

From the inductive hypothesis, the length t prefix of ξ_{t+1}, which is the execution $\xi_t = q_0, \tau, a_t$ in the figure is an $(\delta_0, \varepsilon, \delta_t)$-representative execution. We note that the length t prefix ξ'' is (δ_0, ε)-close to ξ_t. Therefore, $|q_t.X -$

$q''_t.X| \leq \delta_t$. Using the discrepancy function of action a, we have

$$|q_{t+1}.X - q''_{t+1}.X| \leq \beta_a(|q_t.X - q''_t.X|) \leq \beta_a(\delta_t). \tag{4.8}$$

We will quantify the distance between ξ' and ξ''. There are two cases:
(i) If $k = t$ and line 7 is executed. Then, $len(\eta) \leq t - k = 0$, that is, η is an empty string. Hence, ξ' and ξ'' are indeed identical and $q'_{t+1} = q''_{t+1}$. Thus from Equation (4.8),

$$|q_{t+1}.X - q'_{t+1}.X| = |q_{t+1}.X - q''_{t+1}.X| \leq \beta_a(\delta_t).$$

Therefore if δ_{t+1} is computed by line 7, the lemma holds for the length $t+1$ prefix of ξ. (ii) Otherwise $k < 0$ and line 9 is executed. From Lemma 4.5, we can bound the distance between ξ' and ξ'' as

$$|q'_{t+1}.X - q''_{t+1}.X| \leq \gamma_{len(\eta)-1}(\varepsilon) \leq \gamma_{t-k-1}(\varepsilon).$$

Combining with Equation (4.8), we get

$$|q_{t+1}.X - q'_{t+1}.X| \leq |q_{t+1}.X - q'_{t+1}.X| + |q_{t+1}.X - q'_{t+1}.X| \leq \beta_a(\delta_t) + \gamma_{t-k-1}(\varepsilon).$$

Therefore, if δ_{t+1} is computed by line 9, the lemma holds for the length $t+1$ prefix of ξ, which completes the proof. □

Using this lemma, we immediately establish the soundness of the algorithm as follows.

Theorem 4.9. *For any execution ξ' such that ξ' is (δ_0, ε)-close to ξ, the last state of ξ' is in the returned set $\mathcal{B}_{\delta_T}(\xi(T))$.*

Proof. From Lemma 4.8, ξ is a $(\delta_0, \varepsilon, \delta_T)$-representative potential execution. From Definition 4.6, for any potential execution ξ' that is (δ_0, ε)-close to ξ, we have $\xi'.\mathsf{lstate} \in \mathcal{B}_{\delta_T}(\xi(T))$. □

In the following theorem, we show that the radius δ_T can be made arbitrarily small for small enough initial radius δ_0 and the approximation parameter ε.

Theorem 4.10. *For any $t \leq len(\xi)$, as $\delta_0 \to 0$ and $\varepsilon \to 0$, the size of radius δ_T goes to 0.*

Proof. From Proposition 4.4, for any n, $\gamma_n(\varepsilon) \to 0$ as $\varepsilon \to 0$. From Definition 4.2, for any δ_t and discrepancy function β, $\beta(\delta_t) \to 0$ as $\delta_t \to 0$. Therefore, either line 6 or line 8 in Algorithm 4.2 is executed, $\delta_{t+1} \to 0$ as $\delta_t \to 0$ and $\varepsilon \to 0$. Iteratively applying this observation leads that, $\delta_T \to 0$ as $\delta_0 \to 0$ and $\varepsilon \to 0$. Therefore the theorem holds. □

For an execution ξ of length n, the worst case time complexity of Algorithm 4.2 is $O(n^3)$, since finding the anchor position can take at most $O(n^2)$ time. However, if all actions in ξ are mutually approximately independent, the algorithm can over-approximate the final states of a number of $O(n!)$ executions in the best case, which can lead to a dramatical saving in reach set computation. In the coming section, we will use Algorithm 4.2 as a subroutine to compute an over-approximation of the reach set of a labeled transition system.

4.7 Reach Set Over-Approximation

In Section 4.6, we presented a function *generalize*() to over-approximate the reach set of a group of (δ, ε)-close potential executions using a single potential execution. Building upon this function, we will present an algorithm to compute an over-approximation of the reach set of a labeled transition system at a time T.

4.7.1 Equivalent Action Sequences

As we defined in Section 4.3.1, for any set of initial states $S \subseteq \Theta$ and time bound $T \geq 0$, $\mathsf{Execs}(S, T)$ is the set of executions from S with length T, and $\mathsf{Traces}(S, T)$ is the set of action sequences taken by these executions. In concurrent systems, even with S being a singleton $\{q\}$, the size of action sequences $\mathsf{Traces}(q, T)$ can grow exponentially with T. Partial order reduction methods are developed to reduce the number of action sequences needed to be explored. These methods exploit the equivalence relation on the set of action sequences and divide $\mathsf{Traces}(S, T)$ into equivalence classes. Then, checking a representative action sequence suffices to cover the whole equivalence class. Here we define the set of representative action sequences in a

formal way.

Definition 4.8. *Fix any time bound $T \geq 0$, any equivalence relation $\overset{\varepsilon}{\equiv}$ on the set of length T action sequence A^T and any set of state $S \subseteq Q$. The* T-representative action sequences *from S is a set $\mathcal{T} \subseteq \mathsf{Traces}(S, T)$ such that*

(i) any pair of action sequences in $\mathcal{T}$ are not ε-independent, and

(ii) an action sequence $\tau \in \mathcal{T}$ is a T-representative sequence iff there exists an action sequence $\tau' \in \mathsf{Traces}(S, T)$ such that $\tau \overset{\varepsilon}{\equiv} \tau'$.

In another word, the T-representative action sequences $\mathcal{T}$ is the quotient space of the set $\mathsf{Traces}(S, T)$ by the equivalence relation $\overset{\varepsilon}{\equiv}$. Over the past few decades, various partial order methods have been proposed to compute the representative action sequences. Roughly, there are two main classes of methods that have been proposed. For each state q, the *ample/persistent set* techniques select an ample set as a subset of enabled actions from q, where the missing actions are independent to those actions in the ample set [23]. Then execution sequences that take actions from the ample sets are the representatives for equivalent classes. In contrast, the *sleep set* techniques decide which action to explore on-the-fly [88]. The technique maintains a sleep set as a set of actions that should not be explored immediately and update the set based on the independence relation along the execution. In our algorithm, we assume that the T-representative action sequences can be computed for any compact initial set S.

4.7.2 Reach Set Over-Approximation

We propose an algorithm to compute an over-approximation of $\mathsf{Reach}(\Theta, T)$. In this algorithm, we first compute a δ-cover Q_0 of the initial set Θ such that $\mathcal{B}_\delta(Q_0) \supseteq \Theta$. Then, in line 4, we compute the T-representative action sequences from each cover $\mathcal{B}_\delta(q_0)$. Then, we use the algorithm *Generalize*() to compute a ball $\mathcal{R}'$ that over-approximate the set of states reached by a group of executions from a cover following equivalent action sequences. Then the algorithm returns a set $\mathcal{R}$ as a union of such balls $\mathcal{R}'$.

We are able to show that, for any $T \geq 0$, the set $\mathcal{R}$ over-approximate the reach set of the system at time T.

Algorithm 4.3 Reachability algorithm to over-approximate $\mathsf{Reach}(\Theta, T)$

1: $Q_0 \leftarrow \delta\text{-}cover(\Theta)$;
2: $\mathcal{R} \leftarrow \emptyset$;
3: **for** $q_0 \in Q_0$ **do**
4: $\quad \mathcal{T} \leftarrow \mathsf{Traces}(\mathcal{B}_\delta(q_0))/\overset{\varepsilon}{\equiv}$;
5: $\quad$ **for** $\tau \in \mathcal{T}$ **do**
6: $\quad\quad \mathcal{R}' \leftarrow Generalization(\xi_{q_0,\tau}, \delta, \varepsilon, \{\beta_a\}_{a\in A})$;
7: $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}'$;
8: $\quad$ **end for**
9: **end for**
10: **return** $\mathcal{R}$;

Theorem 4.11 (Soundness)**.** *For the set $\mathcal{R}$ returned by Algorithm 4.3, we have $\mathcal{R} \supseteq \mathsf{Reach}(\Theta, T)$.*

Proof. Fixed any $\xi \in \mathsf{Execs}(\Theta, T)$, it suffice to show that $\xi(T) \in \mathcal{R}$. Since $\xi(0) \in \Theta$ and Q_0 is a δ-cover, there exists a $q_0 \in Q_0$ such that $\xi(0) \in \mathcal{B}_\delta(q_0)$. For this q_0, let $\mathcal{T}$ be the T-presentative action sequences from $\mathcal{B}_\delta(q_0)$ computed in line 4 . From Definition 4.8, there must be a representative action sequence $\tau \in \mathcal{T}$ such that $\tau \overset{\varepsilon}{\equiv} trace(\xi)$. Since ξ is (δ, ε)-close to $\xi_{q_0,\tau}$, from Theorem 4.9, the $\mathcal{R}'$ computed using $Generalization()$ guarantees that $\xi.\mathsf{lstate} \in \mathcal{R}$. Therefore the theorem holds. □

We are also able to show that, we can compute the over-approximation up to arbitrary precision.

Theorem 4.12 (Completeness)**.** *For any $r > 0$, there exist $\delta, \varepsilon > 0$ such that, the set $\mathcal{R}$ computed by Algorithm 4.3 satisfies $\mathcal{R} \subseteq \mathcal{B}_r(\mathsf{Reach}(\Theta, T))$.*

Proof. Fix arbitrary $r > 0$. The set $\mathcal{R}$ is a union of balls $\mathcal{R}'$ computed in line 6. Fix any such $\mathcal{R}'$ centered at the last state of potential execution $\xi = \xi_{q_0,\tau}$. It suffices to show that $\mathcal{R}'$ satisfies $\mathcal{R}' \subseteq \mathcal{B}_r(\mathsf{Reach}(\Theta, T))$ for small enough δ and ε.

Since $\tau \in \mathcal{T}$ is a T-representative action sequence from the initial set $\mathcal{B}_\delta(q_0)$, from Definition 4.8, there exists an action sequence $\tau' \in \mathsf{Traces}(\mathcal{B}_\delta(q_0), T)$ such that τ and τ' are ε-equivalent. Hence, there is an execution $\xi' \in \mathsf{Execs}(\mathcal{B}_\delta(q_0), T)$ from the ball $\mathcal{B}_\delta(q_0)$ following the action sequence τ'. By the definition of reach set, we have $\xi'.\mathsf{lstate} \in \mathsf{Reach}(\Theta, T)$. On the other hand, ξ' is (δ, ε)-close to the potential execution ξ. From Theorem 4.9, $\xi'.\mathsf{lstate}$ is

in the ball $\mathcal{R}'$. That is, the ball $\mathcal{R}'$ and the reach set $\mathsf{Reach}(\Theta, T)$ intersect at the state $\xi'.\mathsf{lstate}$.

From Theorem 4.10, the radius of $\mathcal{R}'$ can be made arbitrarily small as δ and ε go to 0. We chose small enough δ and ε, such that the radius of $\mathcal{R}'$ is less than $r/2$. Therefore, the ball $\mathcal{R}'$ is contained in the radius r ball of the reach set $\mathcal{B}_r(\mathsf{Reach}(\Theta, T))$. □

In this section, we proposed an algorithm to over-approximate the reach set of label transition systems. In addition, we showed that the computation can be made arbitrarily precise. We will apply our method to verification of a linear transition system and a temperature control system.

4.8 Case Studies

4.8.1 Linear Transition Systems

We will first study the linear transition systems as presented in Example 4.1. In Example 4.3, we constructed an instance of the system with $N = 3$, and showed that $a_0 \overset{\varepsilon}{\sim} a_2$ and $a_1 \overset{\varepsilon}{\sim} a_2$. The instance of the system has three continuous variables and three actions a_0, a_1, a_2. We also showed that $a_0 \overset{\varepsilon}{\sim} a_2$ and $a_1 \overset{\varepsilon}{\sim} a_2$ with $\varepsilon = 0.1$. In Example 4.4 we further presented an execution and its (δ, ε)-close executions.

For this system, we want to prove if the continuous states can converge to a box $[-0.5, 0.5]^3$ in three rounds. We illustrate an execution $\xi = \xi_{q_0,\tau}$ with $q_0.x = [2.5, -3.5, 1.2]$ and $\tau = a_0a_2a_1a_\perp a_2a_1a_0a_\perp a_1a_0a_2a_\perp$ in Figure 4.7 (blue curve). Using Algorithm 4.2, we compute a tube around the execution to over-approximate all (δ, ε)-close potential executions (green curves). To give an example of δ, ε-close potential executions to ξ, we compute a potential execution $\xi' = \xi_{q_0',\tau}$ with $q_0'.x = [2.3, -3.2, 1]$ and $\tau' = a_0a_1a_2a_\perp a_1a_0a_2a_\perp a_2a_1a_0a_\perp$ (red curve). The result validates that ξ' lies in the tube computed using our technique.

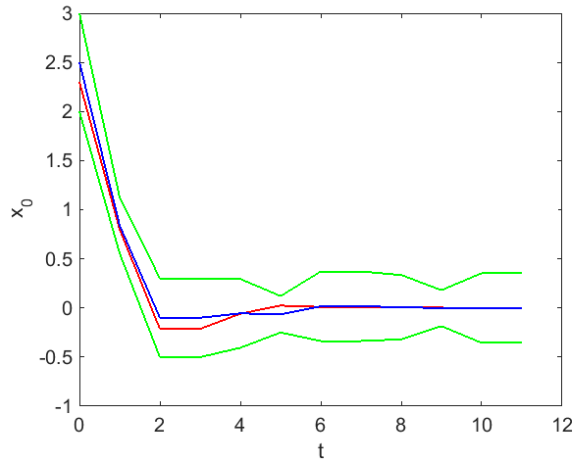


Figure 4.7: A tube around an execution of a linear transition system. The blue curve is the execution ξ. The green curves illustrate a tube around ξ. The red curve is a (δ, ε)-close potential execution with $\delta = 0.5$ and $\varepsilon = 0.1$.

4.8.2 Room Heating Problem

We present a building heating system in Figure 4.8. The building has N rooms each with a heater. For $i \in [N]$, $x_i \in \mathbb{R}$ is the temperature of room i and $m_i \in \{0, 1\}$ captures the off/on state of the heater in the room. The building measures the temperature of rooms periodically every T seconds and save the measurements to y_i. Based on the measurement y_i, each room takes action a_i to decide whether to turn on or turn off its heater. The Boolean variable d_i indicates whether room i has made a decision. These decisions are made asynchronously among the rooms with a small delay h. For this system, we want to check whether the temperature of the room remains in an appropriate range.

```
automata Roomheating(N : Nat)
  variables
    x : Real^N initially x[i] := 60;
    y : Real^N initially y[i] := 60;
    d : Bool^N initially d := false^N;
    m : Bool^N initially m := false^N;

  transitions
    on_i, for i ∈ [N]
      pre  !d_i ∧ y_i <= 72
      eff  x := W(h)x + b(h) + C(h)m;
           d_i := true ∧ m_i := true;

    off_i, for i ∈ [N]
      pre  !d_i ∧ y_i >= 68
      eff  x := W(h)x + b(h) + C(h)m;
           d_i := true ∧ m_i := false;

    flow
      pre ∧_{i∈[N]} d_i
      eff x := W(T)x + b(T) + C(T)m;
           d_i := false for each i ∈ [N];
           y := x;
```

Figure 4.8: Transition system of room heating.

For $i \in [N]$, actions on_i, off_i capture the decision-making process of room i on whether or not to turn on the heater. During the process, time elapses for a (short) period h, which leads to an update of the temperature as an affine function of current temperature x and the heaters state m. The affine function is derived from the thermal differential equation presented in [98]. In this section, we use an instance of the system with the following matrices,

$$W(h) = \begin{bmatrix} 0.96 & 0.01 & 0.01 \\ 0.02 & 0.97 & 0.01 \\ 0 & 0.01 & 0.97 \end{bmatrix}, b(h) = \begin{bmatrix} 1.2 \\ 0 \\ 1.2 \end{bmatrix}, C(h) = \begin{bmatrix} 0.4 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.4 \end{bmatrix}. \quad (4.9)$$

After a room made its decision, either on_i or off_i is taken, it sets the variable d_i to $true$. After all variables d_i are set, action $flow$ captures the time elapse for a (longer) period T after the decision-making of rooms, and updates a measure value y. We use an instance of this step with the following matrices,

$$W(T) = \begin{bmatrix} 0.18 & 0.11 & 0.14 \\ 0.18 & 0.25 & 0.17 \\ 0.09 & 0.13 & 0.28 \end{bmatrix}, b(T) = \begin{bmatrix} 34.2 \\ 24 \\ 30 \end{bmatrix}, C(T) = \begin{bmatrix} 11.4 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 10 \end{bmatrix}. \quad (4.10)$$

For each $i \in [N]$ and $a \in \{on_i, off_i\}$, we will derive the discrepancy function for action a. For any q, q' with $q.L = q'.L$,

$$\begin{aligned} & |a_i(q).x - a_i(q').x| \\ = \; & |W(h)q.x + b(h) + C(h)q.m - W(h)q'.x - b(h) - C(h)q'.m| \\ \leq \; & |W(h)||q.x - q'.x|. \end{aligned}$$

We note that $|W(h)| = 0.99$. Hence, we can define $\beta_a(|q.x - q'.x|) = 0.99|q.x - q'.x|$ as the discrepancy functions of each $a \in \{on_i, off_i\}_{i\in[3]}$. Similarly, we derived that $\beta_{flow}(|q.x - q'.x|) = 0.52|q.x - q'.x|$.

For any $i, j \in [3]$ with $i \neq j$, $a \in \{on_i, off_i\}$ and $b \in \{off_j, off_j\}$, we can prove $a \overset{\varepsilon}{\sim} b$ with $\varepsilon = 0.6$. Notice that, $a_i(q).x = W(h)q.x + b(h) + C(h)q.m =$

$a_j(q).x$ are identical, but $a_i(q).m$ and $a_j(q).m$ could be different.

$$\begin{aligned} & |a_i a_j(q).x - a_j a_i(q).x| \\ = \; & |W(h)a_j(q).x + b(h) + C(h)a_j(q).m - W(h)a_i(q).x - b(h) - C(h)a_i(q).m| \\ = \; & |C(h)a_j(q).m - C(h)a_i(q).m| \leq |C(h)||a_j(q).m - a_i(q).m|. \end{aligned}$$

We note that $|C(h)| = 0.4$. We will give an upper bound on $|a_j(q).m - a_i(q).m|$. Notice that $a_i(q).m$ and $q.m$ can only differ in one bit (m_i). Similarly, $a_j(q).m$ and $q.m$ can only differ in one bit (m_j). Hence $a_i(q).m$ and $a_j(q).m$ can be differ in at most two bits, and $|a_i(q).m - a_j(q)'.m| \leq ||[1, 1, 0]|| = 1.41$. Therefore,

$$|a_i a_j(q).x - a_j a_i(q).x| \leq 0.4 * 1.41 \leq 0.6.$$

Thus for any pair of rooms, the on/off decisions are ε-approximately independent with $\varepsilon = 0.6$. For a round, where each room makes a decision once in arbitrary order, there are in total $3! = 6$ ε-equivalent action sequences.

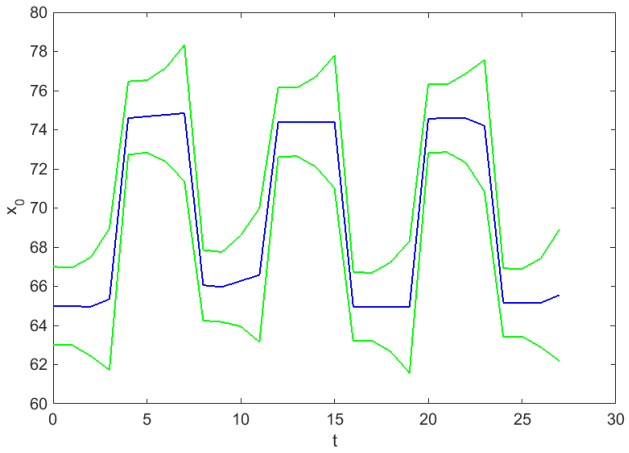


Figure 4.9: A tube around an execution of a room heating problem. The blue curve is the execution ξ. The green curves illustrate a tube around ξ over-approximating (δ, ε)-close potential execution with $\delta = 2$ and $\varepsilon = 0.6$.

We present an execution ξ in Figure 4.9 as the blue curve, which ran for eight rounds. There are a number of 1.6 million (6^8) (δ, ε)-close potential executions of ξ with $\varepsilon = 0.6$ and $\delta = 2$. We illustrate a tube computed by our technique in green, which is proved to contain the final state of all these

(δ, ε)-close potential executions.

4.9 Summary

In this chapter, we propose a partial order reduction for infinite state transition systems. We proposed the notion of ε-independent actions as an extension of the conventional notion of independent actions, such that the resulting states are within ε distance regardless of the order of executing a pair of ε-independent actions. The conventional notion of independent actions is indeed a special case of ε-independent actions with $\varepsilon = 0$. With this ε-independence relation, we are able to define an ε-equivalence relation among traces by swapping consecutive ε-independent actions.

We derive the distance between two executions after inserting a single action at different positions (Lemma 4.5). We observe that the distance between executions depends on the difference in the inserting positions. Based on this result, we proposed an algorithm to compute over-approximated reach set of all executions with ε-equivalent traces. In $O(n^3)$ time, the algorithm can over-approximate a number of at most $O(n!)$ executions. We applied the algorithm to verify properties of a linear transition system and a heater control system.

Chapter 5

DIFFERENTIALLY PRIVATE DISTRIBUTED CONTROL

The *distributed control system* is a class of cyber-physical system, where agents cooperate to achieve individual and global objectives. In these systems, individuals desire to keep certain sensitive data private, while simultaneously sharing some information in order to benefit overall system performance. Hence, there is a dichotomy between privacy and cost. Examples of such systems include peak generation scheduling using consumption data obtained from smart electric meters [39], traffic-aware navigation based on location and destination data obtained from smart GPS [37, 38], and data aggregation from sensor networks [40, 41].

At one extreme is the completely private society where the agents only interact through their coupled dynamics. The other extreme is the completely non-private society. Agents share complete information, which allows all agents to make accurate predictions (e.g., traffic or electricity demands) and to make optimal decisions. Between these two extremes lies a multitude of other possible communication strategies. The privacy-cost trade-off can formalized as the *cost of privacy* measured by the difference between the cost achieved through a given communication strategy and the cost achieved by the completely non-private strategy.

5.1 Privacy-Performance Trade-Off in Distributed Control

In this chapter, we present a general framework for studying cost of privacy for distributed control systems in which a collection of agents pursue individual goals and communicate for the purpose of sensing their shared environment. Each agent i has a *preference* p_i. These preferences capture, for example, a sequence of waypoints for a vehicle or the electric power demand

of a household. The evolution of an agent depends on (a) its dynamics, (b) the control action it takes, and also (c) the environment or the aggregate state of the other agents. If the communication strategy shares more information about agents' preferences, then all agents in the society can estimate the environment more accurately and, therefore, make better control decisions. On the other hand, such a communication strategy may leak information about agent preference. The *cost* of an agent is defined by the deviation of its trajectory from its preference.

In our formulation, once the underlying dynamics of the system, the individual preferences, and the communication strategy are fixed the overall system is deterministic.[1] In our formulation, we show (Proposition 5.1) that knowing the preference for the agents and an observation of the system allows an adversary to uniquely infer the complete state trajectory of an agent over time. The inference process of the adversary is indeed summarized as an *inverse observation mapping* (η^{-1}), which associate the observation sequence with a trajectory of agents.

To keep the sensitive data private, one common approach is to add noise to the communicated information. The effectiveness of such an approach can be measured by using the concept of ε-differential privacy which was first introduced in studies on statistical data bases [101, 43, 44, 45]. Roughly, ε-differential privacy ensures that the probability distribution of any observation does not change substantially with a change in the sensitive data corresponding to any *one* agent. Hence, an adversary cannot infer the sensitive data of agents from the observation. Differentially private algorithms are developed for mechanism design [102], data mining [103] and machine learning [104]. More recently, this notion of privacy is extended to dynamical systems [46, 105] and find its application in optimization [106, 107, 108] and consensus [46, 109, 110].

During the past decade, several varieties of differential privacy have been proposed [101, 44, 46, 111, 112]. The definition of differential privacy used in this chapter (Definition 5.3) is introduced in [112] which augmented the most common definition of differential privacy [101, 44] with metrics. The main technical adjustment to the conventional definition (see e.g. Definition 1 of [43]) is the introduction of a notion of distance on the continuous space

[1] If the communication strategy uses randomization, the the overall system is purely probabilistic.

of sensitive data. A consequence of the generalization is that greater changes in the sensitive data of an agent now permit greater differences between the corresponding probability distributions of observation sequences.

In this chapter, we develop a communication strategy for distributed control systems, which guarantees differential privacy. The main technical step in our approach is to compute the *sensitivity* of a distributed control system. Roughly, the sensitivity of a distributed control system captures the distance between its trajectories with change in one agent's preference. Then, the noise to add follows Laplace distributions parameterized by the sensitivity. We prove that the resulting distributed control system is differentially private. We study the privacy-performance trade-off for linear distributed control systems. We show that the sensitivity of a distributed control system decreases with the stability of its dynamics. Hence, the required standard deviation of noise decreases with the stability of the dynamics. We establish that the cost of differential privacy using our communication strategy up to time T for a system with N agents is at most $O(\frac{T^3}{N\varepsilon^2})$ for stable systems. The cost can also grow exponentially with T for unstable systems. This suggests that the proposed strategy is more likely to be useful for stable systems with short-lived participants (e.g., drivers with short commutes), and further research is needed for strategies that scale better with time.

The rest of this chapter is organized as followng. Section 5.2 discusses related research. Section 5.3 introduces the mathematical formalism we will use throughout the chapter, namely, Markov chains with observations. A Markov chain is essentially a transition system, like the one we studied in Chapter 4, but associated with a probability space. Section 5.5 presents the Laplace communication mechanism for general distributed control systems and Section 5.6 develops specialized results for linear systems.

5.2 Related Works

Several varieties of differential privacy has been proposed in literature [101, 44, 46, 111, 112]. The common requirements for differential privacy is that the change of an individual agent's data can only result in unsubstantial changes in the output distribution. Various mechanisms for achieving differential privacy have been studied in the literature [102, 113, 114]. The

Laplace mechanism requires adding Laplace noise to the query output and was proposed in [42]. To satisfy the privacy condition, the distribution of the Laplace noise is a function of the sensitivity of the query, which is defined as the distance between output if an individual agent changes its data. Then an exponential mechanism, as a generalization of Laplace mechanism, is introduced to design a privacy-preserving auction [102]. To further improve the accuracy of the mechanism measured by the second moment of the noise, one can add a carefully designed noise, whose probability density is a staircase function, instead of standard Laplace noise [115].

In [44], the notion of differential privacy is expanded to include streaming and online computations in which the adversary can look at the entire sequence of outputs from the analysis algorithm. In parallel, with distance defined on the space of data sets, one can measure how much the data have changed and design mechanisms with lower error bounds [116, 117]. Our definition of differential privacy absorbs the extension of the above two lines of work, which ensures that the streaming output distribution does not change substantially if one agent changes its data with a small amount.

Recently, privacy-preserving mechanisms are developed for dynamical systems. In [46], we introduce the notion of differential privacy to a consensus problem and present a Laplace mechanism for solving it. Later, the authors of [110] present an algorithm for unbiased average consensus and quantify the convergence rate. In [118], we proved that adding Laplace noise achieves a minimized entropy of estimation among all differentially private mechanisms. In [119], we used a similar technique to solve a privacy-preserving distributed optimization problem where participants cooperate to minimize the sum of individual costs without exchanging the exact individual cost functions.

Contemporaneously with our work, the authors of [120] adopt the notion of differential privacy to dynamical systems, where an adversary cannot tell the exact input by looking at its output stream. Laplace and Gaussian mechanisms are presented for converting an ordinary dynamical system to a differentially private one. Then, a Kalman filter is designed to estimate the state of differentially private systems with minimized estimation error. The sufficient condition of the minimization problem is established in the form of linear matrix inequalities. This technique is applied to estimate traffic flows with GPS data [121].

5.3 Distributed Control System

In this section, we present a modeling framework for distributed control systems. Like the labeled transitions system we discussed in Chapter 4, the distributed control systems evolve through discrete transitions. However, in contrast, here the transitions are associated with a probability distribution.

5.3.1 Continuous-State Markov Chain with Observations

The underlying model of a distributed control system is a *Markov chain* with possibly uncountably many states.

Definition 5.1. *A* Markov chain *is a tuple* $\langle X \cup Y, \mathcal{F}, \Theta, \kappa \rangle$*, such that*

(i) X *is a set of* internal variables *and* Y *is a set of* observable variables*,* $Q = Val(X \cup Y)$ *is the* state space *equipped with the standard Borel measure* μ *and* $Val(Y)$ *is the* space of observations*,*

(ii) $\mathcal{F} \subseteq 2^Q$ *is a* σ*-algebra on the state space* Q*,*

(iii) $\Theta \in \mathcal{F}$ *is a* set of initial state*,*

(iv) $\kappa : Q \times \mathcal{F} \to [0, 1]$ *is the* kernel function *such that (a) for any* $q \in Q$*,* $\kappa(q, \cdot)$ *is a probability measure on the measurable space* $\langle Q, \mathcal{F} \rangle$*, and (b) for any* $S \in \mathcal{F}$*,* $\kappa(\cdot, S)$ *is a measurable function.*

The variables in X can be either real-valued or finite-valued. For any state q and a measurable set of states S, the kernel of the Markov chain $\kappa(q, S)$ specifies the probability of transition from state q to any state in the set S. We assume the kernel κ is differentiable in the second argument with respect to μ, that is there exists a measurable function $p : S \times S \to \mathbb{R}_{\geq 0}$ such that $\kappa(q, S) = \int_{s \in S} p(q, s) d\mu$ for any $q \in Q$ and $S \in \mathcal{F}$. The function p is the *kernel density* of the Markov chain. Given a kernel density p, a Markov kernel κ is uniquely specified. In this chapter, the state space Q are Euclidean spaces and $\mathcal{F}$ is the standard Borel σ-algebra.

A *trajectory* (or execution) of the Markov chain of length k is a sequence of states $\xi = q_0, q_1, \ldots, q_{k-1}$, such that $q_0 \in \Theta$. For any execution $\xi = q_0, q_1, \ldots, q_{k-1}$, the *probability density* of ξ is defined as $f(\xi) = \prod_{i=0}^{k-1} p(q_i, q_{i+1})$. That is, the probability density of an execution is the product of the kernel

density on each step. The set of executions of length k is Q^k. From Section 2.4, the product σ-algebra with measure μ^k are well defined. Hence for any measurable set of executions $S \subseteq Q^k$, the probability of these sets of execution $\mathbb{P}(S) = \int_{\xi \in S} f(\xi) d\mu^k$ is well defined.

For an execution ξ, the restriction of the execution on the set of observable variables Y is denoted as $\eta(\xi) = \xi \downarrow Y$. We denote $\mathcal{O}$ as the σ-algebra of all measurable observations. It can be checked that, the projection function η is measurable. Hence, for a measurable set of observation $O \in \mathcal{O}$, the probability of the Markov chain generating observations in O is well defined as $\mathbb{P}(O) = \mathbb{P}(\eta^{-1}(O))$.

5.3.2 Distributed Control Systems

In this section, we formally introduce distributed control systems. The behavior of the complete system is modeled as a Markov chain parametrized by a quantity p. We begin by defining a distributed control system abstractly (see Figure 5.1); Section 5.6 provides more concrete instantiations of these definitions in terms of linear models. A system consists of N agents operating in a shared environment. Each agent i, $i \in [N]$, has a preference p_i, which captures its initial state and control objectives. The agent's behavior consists of a physical part which evolves according to some deterministic dynamics and a controller which computes the control inputs for the physical dynamics. The agent uses a communication strategy to broadcast some noisy version of its state to the other agents. The broadcasts are noisy to preserve privacy and are used to estimate the state of the environment. These estimates are used by the i's controller for computing the inputs (along side its own state). Fixing the vector of preferences p for all agents, the evolution of the complete system becomes a Markov chain with observations $\mathcal{M}(p)$. The stochasticity arises from the noise values used in the communication strategy of the individual agents.

The Markov chain modeling the distributed control system consists of N agents. Each agent i, $i \in [N]$, has variables X_i, U_i, Y_i which are the *state*, *input*, and *observable* variables of the agent. For all $i \in [N]$, we write $\mathcal{X} = \mathit{Val}(X_i) = \mathit{Val}(Y_i)$ and $\mathcal{U} = \mathit{Val}(U_i)$, respectively as the *state space* and *input space* of individual agents. Let Z be a set of *environmental variables* and $\tilde{Z}$

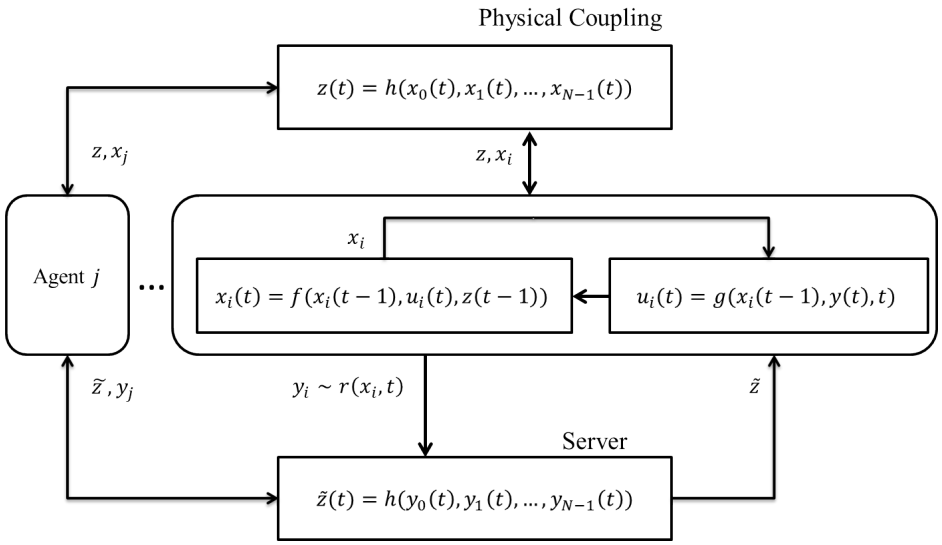


Figure 5.1: Block diagram of a distributed control system.

be the set of *estimated environmental variables* with $Val(Z) = Val(\tilde{Z}) = \mathcal{Z}$. The preferences of each agent i ($p_i \in \mathcal{X}^*$) consisting of a sequence of points in the individual agent's state space $\mathcal{X}$ which defines a path agent i wants to follow. The kernel density of the Markov chain is specified by the following functions: (a) A *dynamics function* $f : \mathcal{X} \times \mathcal{U} \times \mathcal{Z} \to \mathcal{X}$ which defines the next state of an agent as a function of its current state, control input and the environment's state. (b) A *control function* $g : \mathcal{X} \times \mathcal{Z} \times \mathbb{N} \to \mathcal{U}$ which defines the agent's controller output as a function of its state, the estimated environment state. (c) An *aggregation function* $h : \mathcal{X}^N \to \mathcal{Z}$ which defines the state of the environment as a function of the agents' states. And finally (d) A *observation probabilistic density* $r : \mathcal{X} \times \mathcal{X} \times \mathbb{N} \to \mathbb{R}_{\geq 0}$ such that for any $\mathbf{x} \in \mathcal{X}$ and $t \in \mathbb{N}$, $r(\mathbf{x}, \cdot, t)$ is a probability density. The value of $r(\mathbf{x}, \mathbf{y}, t)$ gives the probability density of selecting $\mathbf{y}$ as observable at time t if the actual state is x.

A state of agent i is a point in $\mathcal{X}^2 \times \mathcal{U}$ and its three components are the true agent state (denoted by $\mathbf{x}_i$), the observed agent state ($\mathbf{y}_i$), and the control input ($\mathbf{u}_i$), respectively. The state of the environment is $\mathcal{Z}^2$ and the two components are the (true) environment state ($\mathbf{z}$) and the observed environment state ($\tilde{\mathbf{z}}$). The dynamics functions f and the aggregation function h capture the physical behavior of the system and the coupling between agents—as the control designers, we cannot change them. In this chapter, we assume that the controller function g is obtained through existing control theoretic tech-

niques (see Section 5.2 for a discussion of related work). The only component up for design is the observation density r. In defining the Markov chain below, we will use r to probabilistically update a state component of the agent (called $\tilde{x}_i$ below) which is produced as an observation. This simplifies our model by keeping the observation function η deterministic.

Thus, the state space of the Markov chain modeling the complete system is $Q = (\mathcal{X}^2 \times \mathcal{U})^N \times \mathcal{Z}^2$. For any state $q \in Q$, we write $q.\mathbf{x}_i, q.\mathbf{u}_i$, respectively as the valuations of variables X_i, U_i. We denote $q.\mathbf{x}$ and $q.\mathbf{u}$ as the aggregate vector $\langle q.\mathbf{x}_0, \ldots, q.\mathbf{x}_{N-1}\rangle$ and $\langle q.\mathbf{u}_0, \ldots, q.\mathbf{u}_{N-1}\rangle$, respectively. For a sequence of states $q_0, q_1, \ldots$, we write the corresponding states as $\mathbf{x}(0), \mathbf{x}(1), \ldots$ as the states are clear in the context. The space of observations for the Markov chain is $Y = \mathcal{X}^N \times \mathcal{Z}$. For each $i \in [N]$ we denote the valuations of variables X_i at time $t \in \mathbb{N}$ as $\mathbf{x}_i(t)$. The valuations $\mathbf{u}_i(t), \mathbf{y}_i(t), \mathbf{z}(t), \tilde{\mathbf{z}}(t)$ are defined similarly. The kernel density of the Markov chain at time $t \in \mathbb{N}$ is defined by the following sequence of equations:

$$\begin{aligned}
\mathbf{u}_i(t) &= g(\mathbf{x}_i(t-1), \tilde{\mathbf{z}}(t-1)), && (5.1)\\
\mathbf{x}_i(t) &= f(\mathbf{x}_i(t-1), \mathbf{u}_i(t), \mathbf{z}(t-1)), && (5.2)\\
\mathbf{y}_i(t) &\sim r(\mathbf{x}_i(t), \cdot, t), && (5.3)\\
\mathbf{z}(t) &= h(\mathbf{x}_0(t), \ldots, \mathbf{x}_{N-1}(t)), && (5.4)\\
\tilde{\mathbf{z}}(t) &= h(\mathbf{y}_0(t), \ldots, \mathbf{y}_{N-1}(t)). && (5.5)
\end{aligned}$$

The first three equations define values of the control input ($\mathbf{u}_i$), the agents state ($\mathbf{x}_i$), and the environment state ($\mathbf{z}_i$), for each $i \in [N]$. The value of $\mathbf{y}_i(t)$'s is chosen according to the probability distribution $r(\mathbf{x}_i(t), \cdot, t)$.

The observation function η of the Markov chain is defined as follows: for any state $q \in Q$,

$$\eta(q) = \langle \mathbf{y}_0, \ldots, \mathbf{y}_{N-1}, \tilde{\mathbf{z}}\rangle.$$

In other words, an observation is simply the projection of the state on the Y and $\tilde{Z}$ components. The initial state q_0 is specified by the global preference vector p and the aggregate function h. For each agent, the initial state is defined by the first point of its preference $\mathbf{x}_i(0) = p_i(0)$. Then the aggregate state is $\mathbf{z}(0) = h(\mathbf{x}_0(0), \ldots, \mathbf{x}_{N-1}(0))$. The initial control inputs ($\mathbf{u}_i(0)$) and the initial observed agent state ($\mathbf{y}_i(0)$) and initial observed environment state ($\tilde{\mathbf{z}}(0)$) are set to 0. We denote $\mathsf{Execs}(p)$ as the set of all executions of $\mathcal{M}(p)$.

The only sources of uncertainty in the behavior of a control system are (a) the preferences of the agents (p) and (b) the randomized observation map (r), which is used to disseminate noisy private information for the sake of better performance. Thus, given a preference vector and an observation sequence, and the knowledge of the parameters f, g, h, it is possible to infer a unique execution of $\mathcal{M}(p)$. We formalize this notion in Proposition 5.1.

Proposition 5.1. *For any distributed control system $\mathcal{M}(p)$, given a preference vector p and an observation sequence β of length k, $\eta^{-1}_{\mathcal{M}(p)}(\beta)$ is a singleton set.*

Proof. The proof is by induction on the length of β.
Base. If β is of length one then $\eta^{-1}(\beta)$ is the single starting state θ. As we mentioned previously, the agent i's state matches the first point of p_i, that is $\mathbf{x}_i(0) = p_i(0)$, which is specified by p. Also, $\mathbf{z}(0) = h(\mathbf{x}_0(0), \ldots, \mathbf{x}_{N-1}(0))$. And other variables are initialized as 0. Thus, the start state θ is fixed.
Induction. Suppose $\beta = \beta' b$ be an observation of length $k+1$, where $\eta^{-1}(\beta')$ is the unique execution ξ' ending with last state $q_k = \langle \mathbf{u}(k), \mathbf{x}(k), \mathbf{y}(k), \mathbf{z}(k), \tilde{\mathbf{z}(k)} \rangle$. It suffices to show that for the given state q_k and the observation y, there is a unique state q_{k+1} which makes $\xi' q_{k+1} = \eta^{-1}(\beta)$. From Equation (5.1), it follows that for each $i \in [N]$, $\mathbf{u}_i(k+1)$ is uniquely defined as $g(\mathbf{x}_i(k), \tilde{\mathbf{z}}(k))$. This and Equation (5.2) imply that $\mathbf{x}_i(k+1)$ is uniquely defined. Similarly, the $\mathbf{u}_i$'s and the $\mathbf{x}_i$'s together with Equation (5.4) imply that $\mathbf{z}(k+1)$ is also uniquely defined. Finally, $\mathbf{y}_i(k+1)$ and $\tilde{\mathbf{z}}(k+1)$ are specified by the last state b of β. Hence, the states at $k+1$ step are fully specified. □

Thus, fixed an observation sequence β, an execution ξ is fully specified by a global preference p. Fix an observation sequence, if the adversary has a high confident guess of the preference vector p, then the agents' whole trajectory corresponding to such a p is also of high validity. Otherwise, if the preference vector is protected, the evolution of the agents is hidden. So it suffices to consider privacy of the preference vector p.

5.4 Privacy and Cost in Distributed Control

For formulating privacy of a distributed control system, we first define comparable and adjacent preference vectors. For a pair of preference vectors p and p', the corresponding Markov chains $\mathcal{M}(p)$ and $\mathcal{M}(p')$ are *comparable* if the observable spaces are identical, that is, $\mathcal{O}_{\mathcal{M}(p)} = \mathcal{O}_{\mathcal{M}(p')}$.

Definition 5.2. *A pair of preference vectors p and p' in $(\mathcal{X}^N)^*$ are* adjacent up to time T, *written as T-adj(p, p') in short, if there exists a $k \in [N]$, such that for all $t \leq T$, such that (i) $|p_k(t) - p'_k(t)| \leq 1$, and (ii) for all $i \neq k$ $p_i(t) = p'_i(t)$.*

The norm used in the definition is the standard ℓ^p-norm for arbitrary $p \in [1, \infty]$ chosen by the user. In other words, two preference vectors are T-adjacent if they differ only in the preferences of a single agent up to time T, and the difference in terms of the ℓ^p-norm is at most unity at each time. We adapt the standard definition of differential privacy to this framework of control mechanisms, where the protection of the individual agent's preferences have to be balanced with the benefits of information sharing for control in a shared environment.

Definition 5.3. *The randomized control mechanism $\mathcal{M}$ is* ε-differentially private *up to time T, if for any two T-adjacent preference vectors p and p' and any set of finite observation sequences Obs, $\mathcal{M}(p)$ and $\mathcal{M}(p')$ are comparable and*

$$\mathbb{P}_{\mathcal{M}(p)}[\eta^{-1}_{\mathcal{M}(p)}(Obs)] \leq e^{\varepsilon}\mathbb{P}_{\mathcal{M}(p')}[\eta^{-1}_{\mathcal{M}(p')}(Obs)]. \tag{5.6}$$

This definition of differential privacy is similar to the one appears in [44] with two technical differences. First, we restrict the preferences to be adjacent up to a time bound. Secondly, owing our choice of the definition of $\mathcal{M}(p)$ which allows all the components of $\mathcal{M}(p)$ to possibly depend on p, for privacy of individual agent's preferences with respect to observation sequences produced from to Markov chains, it is required that the output spaces of the corresponding chains are the same. This requirement is incorporated by making the chains comparable.

Performance of a distributed control system is measured by a cost function. It is standard to consider the following quadratic cost in optimal control

theory. Given an execution of length $T+1$, $\alpha = q_0, q_1, \ldots, q_T$, the cost of control for an individual agent i up to T time is the sum of squared distance between the agent's state and its preferred state. That is, $cost_{\mathcal{M}(p),i}(\alpha) \triangleq \sum_{t=1}^{T} |\mathbf{x}_i(q_t) - p_i(t))|_2^2$. The summation starts with $t = 1$ because by definition $x_i(q_0) = p_i(0)$ and no cost is paid at time $t = 0$. The cost function of agent i is the expectation of the function $cost_{\mathcal{M}(p),i}(\alpha)$ over the space of executions of length T,

$$cost_{\mathcal{M}(p),i}(T) = \mathbb{E}\left[\sum_{t=1}^{T} |\mathbf{x}_i(q_t) - p_i(t))|_2^2\right].$$

We give an example of a distributed control problem.

Example 5.1 (Private Nevigation). This example captures the routing of N agents on a 2-D plane whose motion is affected by the center of gravity of all the agents. The agent i's state $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^2$ has two components, which are the x and y coordinates of agent i. Each agent has a preference which is a path $p_i \in (\mathbb{R}^2)^*$. The individual agent's state at time t is affected by three factors: the previous state $\mathbf{x}_i(t-1)$, the aggregate state, which is the center of the mass of the herd ($\mathbf{z}(t-1)$), and the individual's control input $\mathbf{u}_i(t)$. The update law of the ith agent's state at time $t+1$ follows:

$$\mathbf{x}_i(t) = 1.5\mathbf{x}_i(t-1) + c\mathbf{z}(t-1) + \mathbf{u}_i(t). \tag{5.7}$$

The aggregate state $z \in \mathcal{Z} \subset \mathbb{R}^2$ is the center of gravity of the herd,

$$z(t) = \frac{1}{N}\sum_{i\in[N]} x_i(t).$$

Designing a controller for the i-th agent, which cancels out the influence of the aggregate state on individual agent and drives it toward the goal $p_i(t+1)$, needs the actual states of all other agents. With the precise information of others, agent i can achieve its desirable individual cost by using some optimal control technique. For example, the following controller may be used:

$$\mathbf{u}_i(t) = -c\mathbf{z}(t-1) - 1.3\mathbf{x}_i(t-1) + 0.8p_i(t). \tag{5.8}$$

Combined Equations (5.8) and (5.7), we get the update rule for the whole close-loop system:

$$\mathbf{x}_i(t) = 0.2\mathbf{x}_i(t-1) + 0.8p_i(t). \tag{5.9}$$

The current state $\mathbf{x}_i(t)$ is a linear combination of the previous state $\mathbf{x}_i(t-1)$ and the current preference $p_i(t)$. If the sequence of preference p_i is fixed for a few rounds, the state $\mathbf{x}_i$ converges to it geometrically. Otherwise if the sequence of preference is changing, the state $\mathbf{x}_i$ keeps tracking it. The cost of the individual agent is defined by the sum of squared distance between its state $\mathbf{x}_i$ and p_i, that is,

$$cost(p, T) = \sum_{t=1}^{T} |x_i(t) - p_i(t)|_2^2.$$

In Section 5.6, we introduce a mechanism that guarantees differential privacy for this example.

We define the cost of a randomized control mechanism $\mathcal{M}$ as the difference in the cost of two nearly identical Markov chains with observations $\mathcal{M}(p)$ and $\mathcal{M}'(p)$, where $\mathcal{M}(p)$ is the differentially private Markov chain and $\mathcal{M}'(p)$ is identical except that its observation map discloses perfect information about the agents' states. Formally, given $\mathcal{M}(p)$ defined by the parameters f, g, h, and r, the perfectly observable version $\mathcal{M}'(p)$ is defined by the parameters f, g, h and r', where for any $t \in \mathbb{N}$, $r'(x, y, t) = \delta(y - x)$ with $\delta(\cdot)$ being the Dirac delta distribution. Then, the cost of privacy is defined as the difference in the costs of $\mathcal{M}(p)$ and $\mathcal{M}'(p)$.

Definition 5.4. *For any $\varepsilon > 0$ and time bound $T \in \mathbb{N}$, and an ε-differentially private randomized control mechanism $\mathcal{M}$, the* Cost of Privacy (CoP) *up to time T, is defined by the supremum of the difference between any individual's cost in $\mathcal{M}(p)$ and the corresponding perfectly observable chain $\mathcal{M}'(p)$ over all preference vector p:*

$$CoP(\varepsilon, \mathcal{M}, T) = \sup_{p,i} \left(cost_{\mathcal{M}'(p),i}(T) - cost_{\mathcal{M}(p),i}(T) \right).$$

We will discuss the cost of privacy of Example 5.1 in Section 5.6.2

5.5 Laplace Observations of Differential Privacy

In this section, we introduce a strategy for creating observation maps that guarantees differential privacy of the agents' preferences. For the remainder

of this chapter let $n = |X_i|$ be the length of local state $\mathbf{x}_i$. In this design, at time t, each agent reports $\mathbf{y}_i(t)$ by adding a noise $\omega_i(t)$ on its actual state $\mathbf{x}_i(t)$, that is

$$\mathbf{y}_i(t) = \mathbf{x}_i(t) + \omega_i(t), \tag{5.10}$$

where $\omega_i(t)$ is a vector that consists of n independent random noises drawn from Laplace distribution $Lap(M_t)$. If we combine the two step, we have

$$\mathbf{y}(t) = h(\mathbf{y}(t)) = h(\mathbf{x}(t) + \omega(t)). \tag{5.11}$$

For brevity, we denote $\mathbf{x}(t)$ and $\omega(t)$ as the global state and the global noise vector.

Before proposing an actual design of M_t, we first define the sensitivity of the system. Fix an observation sequence $\beta \in Y^T$ up to time T and a preference vector p. As we mentioned in Proposition 5.1, $\eta^{-1}_{\mathcal{M}(p)}(\beta)$ is a singleton set. Then, $\mathbf{x}(\eta^{-1}_{\mathcal{M}(p)}(\beta)(t))$ is the global state at time t corresponding to the execution $\eta^{-1}_{\mathcal{M}(p)}(\beta)$. To quantify the maximal difference of the system's global state at time t resulted from a pair of adjacent preference vectors p and p', we introduce the sensitivity of the system.

Definition 5.5. *For a mechanism $\mathcal{M}$, we define the sensitivity of $\mathcal{M}$ at time $t \in \mathbb{N}$ as*

$$\Delta(t) = \sup_{\beta \in \mathcal{O}^t} \sup_{t\text{-}adj(p,p')} |\mathbf{x}(\eta^{-1}_{\mathcal{M}(p)}(\beta(t))) - \mathbf{x}(\eta^{-1}_{\mathcal{M}(p')}(\beta(t)))|_1.$$

Sensitivity captures the L^1 distance between executions with adjacent preference vectors. We assume that $\Delta(t)$ is bounded for any $t \in \mathbb{N}$ throughout the chapter. The sensitivity can be computed explicitly for distributed control systems with linear dynamics and control, and can be estimated numerically for nonlinear systems. Using the sensitivity, we develop a noise-adding strategy to achieve differential privacy.

Theorem 5.2. *At each time $t \in [T]$, if each agent adds a noise vector $\omega_i(t)$ which consists of n independent Laplace noise $Lap(M_t)$ such that $\sum_{t=0}^{T} \frac{\Delta(t)}{M_t} \leq \varepsilon$, then the distributed control system is ε-differentially private up to time T.*

Proof. Fix any pair of T-adjacent preference vectors $p, p' \in \mathcal{X}^{TN}$, and any set of observation sequence $Obs \subseteq \mathcal{O}^T$. We will denote the sets of executions

$\eta^{-1}_{\mathcal{M}(p)}(Obs)$ and $\eta^{-1}_{\mathcal{M}(p')}(Obs)$ by A and A', respectively. First, we define a correspondence B between the sets A and A'. For $\xi \in A$ and $\xi' \in A'$, $B(\xi) = \xi'$ if and only if they are the observation sequence up to time T. That is, $\eta(\xi(t)) = \eta(\xi'(t))$ for all $t \in [T]$. From Proposition 5.1, for any observation sequence $\beta \in Obs$ there is a unique execution $\xi \in \mathsf{Execs}(p)$ that can produce the observation. Similarly, ξ' is also unique in $\mathsf{Execs}(p')$. So B is indeed a bijection. We relate the probability measures of the sets of executions A and A',

$$\frac{\mathbb{P}_{\mathcal{M}(p)}[\eta^{-1}_{\mathcal{M}(p)}(Obs)]}{\mathbb{P}_{\mathcal{M}(p')}[\eta^{-1}_{\mathcal{M}(p')}(Obs)]} = \frac{\int_{\xi \in A} \mathbb{P}_{\mathcal{M}(p)}[\xi] d\mu}{\int_{\xi' \in A'} \mathbb{P}_{\mathcal{M}(p')}[\xi'] d\mu'}. \tag{5.12}$$

Changing the variable using the bijection B we have,

$$\begin{aligned} \int_{\xi' \in A'} \mathbb{P}_{\mathcal{M}(p')}[\xi'] d\mu' &= \int_{B(\xi) \in A'} \mathbb{P}_{\mathcal{M}(p)}[B(\xi)] d\mu \\ &= \int_{\xi \in A} \mathbb{P}_{\mathcal{M}(p)}[B(\xi)] d\mu. \end{aligned} \tag{5.13}$$

From Equations (5.1)-(5.5) and the definition of r,

$$\int_{\xi \in A} \mathbb{P}_{\mathcal{M}(p)}[\xi] d\mu = \int_{\xi \in A} \mathbb{P}_{\mathcal{M}(p)}[\xi.\mathbf{y} | \xi.\mathbf{x}] d\mu,$$

where $\mathbf{x}(t)$ is the vector of N agents' states at t along execution ξ. Each $\mathbf{x}_i(t)$ is a vector of length n. We denote the k-th state component of $\mathbf{x}_i(t)$ by $\mathbf{x}_i^{(k)}(t)$. As $\mathbf{y}(t)$ is obtained by adding $n \times N$ independent noise values to $\mathbf{x}(t)$, from the distribution $Lap(M_t)$, it follows that the probability density of an execution is reduced to

$$\mathbb{P}_{\mathcal{M}(p)}[\mathbf{y}(\xi) | \mathbf{x}(\xi)] = \prod_{\substack{i \in [N], k \in [n] \\ t \in [T]}} p_L(\xi(t).\mathbf{y}_i^{(k)} - \xi(t).\mathbf{x}_i^{(k)} | M_t), \tag{5.14}$$

where $p_L(x|b)$ is the probability density function at x with parameter b. Then, we relate the distance at time t between the states of ξ and $B(\xi)$ with the sensitivity $\Delta(t)$. Let $\beta = \eta(\xi)$ be the observation sequence corresponding to ξ. By the Definition 5.5, we have

$$|\xi(t).\mathbf{x} - \xi'(t).\mathbf{x}|_1 \leq \Delta(t).$$

The norm in the above equation is L^1-norm. The global state $x(t)$ consists of N local states $\mathbf{x}_i(t)$, each of which has n component. So $\xi(t).\mathbf{x}$ and $\xi'(t).\mathbf{x}$ lives in space $\mathbb{R}^{nN}$. By definition of L^1-norm:

$$\sum_{i=1}^{N}\sum_{k=1}^{n}|\xi(t).\mathbf{x}_i^{(k)}-\xi'(t).\mathbf{x}_i^{(k)}|=|\xi(t).\mathbf{x}_i-\xi'(t).\mathbf{x}_i|_1\leq\Delta(t).$$

By the definition of bijection B, the observations of ξ and $B(\xi)$ match, that is, $\mathbf{y}(\xi(t))=\mathbf{y}(B(\xi)(t))$. From the property of Laplace distribution,

$$\begin{aligned}
&\prod_{i\in[N]k\in[n]}\frac{p_L(\xi(t).\mathbf{y}_i^{(k)}-\xi(t).\mathbf{x}_i^{(k)}|M_t)}{p_L(B(\xi)(t).\mathbf{y}_i^{(k)}-B(\xi)(t)\mathbf{x}_i^{(k)}|M_t)}\\
\leq&\prod_{i\in[N],k\in[n]}\exp\left(\frac{|\xi(t).\mathbf{y}-\xi(t).\mathbf{x}-B(\xi)(t).\mathbf{y}+B(\xi)(t).\mathbf{x}|}{M_t}\right)\\
=&\prod_{i\in[N],k\in[n]}\exp\left(\frac{|\xi(t).\mathbf{x}(\xi(t))-B(\xi)(t).\mathbf{x}|}{M_t}\right)\\
=&\exp\left(\sum_{i\in[N],k\in[n]}\frac{|\xi(t).\mathbf{x}-B(\xi)(t).\mathbf{x}|}{M_t}\right)\\
\leq&e^{\frac{\Delta(t)}{M_t}}.
\end{aligned}\tag{5.15}$$

Combining Equations (5.12), (5.13), (5.14) and (5.15), we derive

$$\begin{aligned}
&\frac{\mathbb{P}_{\mathcal{M}(p)}[\eta^{-1}_{\mathcal{M}(p)}(Obs)]}{\mathbb{P}_{\mathcal{M}(p')}[\eta^{-1}_{\mathcal{M}(p')}(Obs)]}\\
\leq&\frac{\int_{\xi\in A}\mathbb{P}_{\mathcal{M}(p)}[\xi.\mathbf{y}|\xi.\mathbf{x}]d\mu}{\int_{\xi\in A}\mathbb{P}_{\mathcal{M}(p)}[B(\xi).\mathbf{y}|B(\xi).\mathbf{x}]d\mu}\\
\leq&\prod_{t\in[T]}e^{\frac{\Delta(t)}{M_t}}\leq e^{\sum_{t\in[T]}\frac{\Delta(t)}{M_t}}.
\end{aligned}$$

If M_t satisfy $\sum_{t=0}^{T}\frac{\Delta_D(t)}{M_t}\leq\varepsilon$, then $\prod_{t\in[T]}e^{\frac{\Delta(t)}{M_t}}\leq e^{\varepsilon}$. Thus the theorem holds. □

We can also derive the following corollary from Theorem 5.2.

Corollary 5.3. *At each time $t\in[T]$ if each agent adds an vector of independent Laplace noise $Lap(M_t)$, where $M_t=\frac{\Delta_D(t)T}{\varepsilon}$ to its actual state, then the distributed control system is ε-differentially private.*

In this mechanism, the noise added is proportional to the sensitivity of the system and the time bound of the system T. Roughly, an adversary can examine a number of T observations of an individual agent. The parameter of the Laplace noises added is proportional to the length of the observation and the sensitivity of the system.

5.6 Differentially Private Linear Distributed Control

In this section, we will specialize the general framework of Section 5.3 to linear control systems. Linear models for the physical dynamics and linear controller functions are the predominant models studied in control theory literature. In this setup, the optimal controller design problem can be formulated and solved effectively using convex optimization. We assume that agent i's state ($\mathbf{x}_i$), its observed state ($\mathbf{y}_i$), its control input ($\mathbf{u}_i$), the environment state ($\mathbf{z}$), and the observed environment state ($\tilde{\mathbf{z}}$) are all points in $\mathbb{R}^n$, for some natural number n. Agent i's preference is an infinite (possibly repeated) sequence of points in $\mathbb{R}^n$. Next, we define the remaining four parameters of the control system. The linear dynamics function for the i-th agent is:

$$f(\mathbf{x}_i, \mathbf{z}, \mathbf{u}_i) = A\mathbf{x}_i + c\mathbf{z} + \mathbf{u}_i,$$

where $A \in \mathbb{R}^{n\times n}$ is the dynamics matrix and $c \in \mathbb{R}$ is a coupling constant. The linear aggregation function h computes the average of the agents' states, which is defined as

$$h(\mathbf{x}) = \frac{1}{N}\sum_{i\in[N]} \mathbf{x}_i.$$

For this type of dynamics, a linear feedback controller suffices to drive the agent to any fixed preference point. We choose a general linear feedback control function of the form:

$$g(\mathbf{x}_i, \mathbf{z}, t) = (K - A)\mathbf{x}_i + (I - K)p_i(t) - c\tilde{\mathbf{z}},$$

where $K \in \mathbb{R}^{n \times n}$ is a *stable matrix* and I is the identity matrix. Finally, the form of the observation map is

$$\mathbf{y}_i = \mathbf{x}_i + \omega_i(t),$$

where $\omega_i(t)$ is drawn from a time-dependent probability distribution to be defined below.

As in the general case (Section 5.3), given a preference vector p, the above parameters define the Markov chain $\mathcal{M}(p)$ which captures the evolution of the system. The system of equations defining the transitions of this Markov chain, corresponding to Equations (5.1)-(5.5), can be written as follows: At time $t \in \mathbb{N}$,

$$\mathbf{u}_i(t) = (K - A)\mathbf{x}_i(t-1) + (I - K)p_i(t) - c\tilde{\mathbf{z}}(t-1), \tag{5.16}$$

$$\mathbf{x}_i(t) = A\mathbf{x}_i(t-1) + c\mathbf{z}(t-1) + \mathbf{u}_i(t), \tag{5.17}$$

$$\mathbf{y}_i(t) = \mathbf{x}_i(t) + \omega_i(t), \tag{5.18}$$

$$\mathbf{z}(t) = \frac{1}{N}\sum_{i \in [N]} \mathbf{x}_i(t), \tag{5.19}$$

$$\tilde{\mathbf{z}}(t) = \frac{1}{N}\sum_{i \in [N]} \mathbf{y}_i(t). \tag{5.20}$$

Combining the above equations, the closed-loop dynamics of agent i is:

$$\mathbf{x}_i(t) = K\mathbf{x}_i(t-1) + (I - K)p_i(t) - \frac{c}{N}\sum_{i \in [N]} \omega_i(t-1). \tag{5.21}$$

Agent i's state at time t can be written as a function of its preference sequence $\{p_i(s)\}_{s \le t}$ and the sequence $\{\omega_i(s) : i \in [N], s \le t\}$ of noise vectors added in all previous rounds. By iteratively applying Equation (5.21), we obtain:

$$\begin{aligned} x_i(t) =& K^t p_i(0) + \sum_{s=1}^{t} K^{t-s}(I - K)p_i(s) \\ &- \frac{c}{N}\sum_{s=0}^{t-1} K^{t-s-1} \sum_{i \in [N]} \omega_i(s). \end{aligned} \tag{5.22}$$

Remark 5.1. By taking expectation on both sides of Equation (5.21), we

can write $\mathbf{x}(t) - p(t) = K(\mathbf{x}(t-1) - p(t))$. Given a stable the matrix K, for agent i, after update at time t, the new state gets closer to the preference $p_i(t)$. The more stable K is, the better tracking $x(t)$ performs toward $p(t)$.

For representing the dynamics of the complete system with N agents, we define two $nN \times nN$ matrices

$$\mathbf{K} \triangleq \begin{bmatrix} K & & \\ & \ddots & \\ & & K \end{bmatrix} \text{ and } \mathbf{C} \triangleq \frac{c}{N} \begin{bmatrix} I & \dots & I \\ \vdots & \ddots & \vdots \\ I & \dots & I \end{bmatrix},$$

where $\mathbf{K}$ is a block diagonal matrix with K matrices as its diagonal blocks and $\mathbf{C}$ is a block matrix with all the blocks set to $\frac{c}{N}$ times the identity matrix I. Combining the Equation (5.21) for all the N agents we obtain:

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{K}\mathbf{x}(t-1) + (I - \mathbf{K})p(t) + \mathbf{C}\mathbf{x}(t-1) - \tilde{\mathbf{z}}(t-1) \\ &= (\mathbf{K} + \mathbf{C})\mathbf{x}(t-1) + (I - \mathbf{K})p(t) - \tilde{\mathbf{z}}(t-1). \end{aligned} \tag{5.23}$$

Given a preference vector p and an observation sequence β, by Proposition 5.1, we know that there is a unique execution $\eta^{-1}_{\mathcal{M}(p)}(\beta)$. The vector of agents' states at time $t \geq 0$, along this execution is $\mathbf{x}(\eta^{-1}_{\mathcal{M}(p)}(\beta)(t))$. Iteratively applying Equation (5.23) we obtain:

$$\begin{aligned} \mathbf{x}(\eta^{-1}_{\mathcal{M}(p)}(\beta(t))) = (\mathbf{K} + \mathbf{C})^t p(0) - \sum_{s=0}^{t-1} (\mathbf{K} + \mathbf{C})^{t-s} \tilde{\mathbf{z}}(\beta(t)) \\ + \sum_{s=1}^{t} (\mathbf{K} + \mathbf{C})^{t-s} (I - \mathbf{K}) p(s). \end{aligned}$$

5.6.1 Sensitivity of Linear Distributed Control

In this section, we state Theorem 5.4 which establishes bound on the sensitivity $\Delta(t)$. For proving this theorem, we fix two Markov chains of the system (Equations (5.16)-(5.20)) with adjacent preference vectors p and p' and compute the difference between two chains. Recall that p and p' are identical except the preference of one agent (i). Then, the difference between the two Markov chains has two components: (1) the change in agent i's state, and

(2) the sum of changes in other agents' states. The sensitivity is then computed as a bound of the sum of above two components. With this bound on sensitivity, we introduce a Laplace mechanism defining the observation map (r) in Corollary 5.5 and then show that the mechanism achieves differential privacy of the linear distributed control system.

Theorem 5.4. *For the linear distributed control system, for all $t \in \mathbb{N}$ the sensitivity $\Delta(t)$ is upper-bounded by $\kappa(t)$, where*

$$\kappa(t) \triangleq |G^t - K^t| + |K^t| + |H| \sum_{s=0}^{t-1} (|G^s - K^s| + |K^s|),$$

with $G \triangleq cI + K$ and $H \triangleq I - K$.

Proof. We fix a time t, a pair of t-adjacent preferences p and p', and a sequence of observations $\beta = \langle \tilde{z}(0), \tilde{x}(0) \rangle, \langle \tilde{z}(1), \tilde{x}(1) \rangle \ldots$. Then, by Equation (5.23) we get

$$\begin{aligned} & |\mathbf{x}(\eta^{-1}_{\mathcal{M}(p)}(\beta)(t)) - \mathbf{x}(\eta^{-1}_{\mathcal{M}(p')}(\beta)(t))| \\ = \; & |(\mathbf{K} + \mathbf{C})^t (p(0) - p'(0)) \\ & + \textstyle\sum_{s=1}^{t} (\mathbf{K} + \mathbf{C})^{t-s} (I - \mathbf{K})(p(s) - p'(s))|. \end{aligned} \tag{5.24}$$

Equation (5.24) is independent of the observation sequence because β affects the two executions $\eta^{-1}_{\mathcal{M}(p)}(\beta)$ and $\eta^{-1}_{\mathcal{M}(p')}(\beta)$ identically and cancels out. For the remainder of the proof we write $\mathbf{x}(t)$ and $\mathbf{x}'(t)$ for $\mathbf{x}(\eta^{-1}_{\mathcal{M}(p)}(\beta)(t))$ and $\mathbf{x}(\eta^{-1}_{\mathcal{M}(p')}(\beta)(t))$. It follows from Definition 5.5 that $\Delta(t) = \sup_{T\text{-}adj(p,p')} |\mathbf{x}(t) - \mathbf{x}'(t)|$.

We will first expand the term $(\mathbf{K} + \mathbf{C})^s$ on the right-hand side of Equation (5.24). In block matrix form,

$$(\mathbf{K} + \mathbf{C})^s = \left(\begin{bmatrix} K & & \\ & \ddots & \\ & & K \end{bmatrix} + \frac{c}{N} \begin{bmatrix} I & \ldots & I \\ \vdots & \ddots & \vdots \\ I & \ldots & I \end{bmatrix} \right)^s . \tag{5.25}$$

The matrix $(\mathbf{K} + \mathbf{C})$ has two types of blocks: (1) $K + \frac{c}{N} I$ as the diagonal blocks and (2) $\frac{c}{N} I$ as the off-diagonal blocks. As K and I are commutative,

applying binomial expansion of the Equation (5.25) and after some lengthy but elementary linear algebra the product matrix $(\mathbf{K}+\mathbf{C})^s$ becomes

$$(\mathbf{K}+\mathbf{C})^s = \begin{bmatrix} P_s & Q_s & \dots & Q_s \\ Q_s & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & Q_s \\ Q_s & \dots & Q_s & P_s \end{bmatrix}, \tag{5.26}$$

where $Q_s = \frac{1}{N}(G^s - K^s)$, $P_s = Q_s + K^s$, and $G = cI + K$. From Equation (5.26), we also obtain:

$$(\mathbf{K}+\mathbf{C})^s(I-\mathbf{K}) = \begin{bmatrix} P'_s & Q'_s & \dots & Q'_s \\ Q'_s & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & Q'_s \\ Q'_s & \dots & Q'_s & P'_s \end{bmatrix}, \tag{5.27}$$

where $Q'_s = Q_s H$, $P'_s = Q'_s + K^s H$, and $H = I - K$. With Equations (5.26) and (5.27), we bound the right-hand side of Equation (5.24). Recall that from Definition 5.2, $t\text{-}adj(p, p')$ if and only if there exists some $i \in [N]$, for all $s \leq t$, (a) $|p_i(s) - p'_i(s)| \leq 1$, and (b) for all $j \neq i$, $p_j(s) - p'_j(s) = 0$. That is, for any $s \leq t$,

$$p(s) - p'(s) = \begin{bmatrix} 0, & \dots, 0, [p_i(s) - p'_i(s)]^\top, 0, \dots, 0 \end{bmatrix}^\top, \tag{5.28}$$

has n non-zero entries corresponding to the preferences of some agent i, and all other entires are 0. Then, $(\mathbf{K}+\mathbf{C})^s(p(s)-p'(s))$ is a vector, where the i-th component is $P_s(p_i(s) - p'_i(s))$ and other components are $Q_s(p_i(s) - p'_i(s))$. Similarly $|(\mathbf{K}+\mathbf{C})^s(I-K)(p(s)-p'(s))|$ is a vector, where the i-th component is $P'_s(p_i(s) - p'_i(s))$ and other components are $Q'_s(p_i(s) - p'_i(s))$. Therefore, the term inside the norm on the right-hand side of Equation (5.24) is a vector where the i-th component is

$$P_t(p_i(0) - p'_i(0)) + \sum_{s=1}^{t} P'_{t-s}(p_i(s) - p'_i(s)), \tag{5.29}$$

and all the other $N-1$ components are

$$Q_t(p_i(0) - p'_i(0)) + \sum_{s=1}^{t} Q'_{t-s}(p_i(s) - p'_i(s)). \tag{5.30}$$

Substituting Equations (5.29) and (5.30) into Equation (5.24), combining with $|(p_i(s) - p'_i(s)| \leq 1$, we have

$$|\mathbf{x}(t) - \mathbf{x}'(t)| \leq (N-1)(|Q_t| + \sum_{s=0}^{t-1} |Q'_s|) + |P_t| + \sum_{s=1}^{t} |P'_s|. \tag{5.31}$$

It follows that $\Delta(t) \leq (N-1)(|Q_t| + \sum_{s=0}^{t-1} |Q'_s|) + |P_t| + \sum_{s=1}^{t} |P'_s|$. Using Equations (5.26) and (5.27), we represent P_s, P'_s by Q_s, Q'_s, K and H. Therefore,

$$\begin{aligned} & (N-1)(|Q_t| + \textstyle\sum_{s=0}^{t-1} |Q'_s|) + |P_t| + \sum_{s=1}^{t} |P'_s| \\ \leq\ & (N-1)(|Q_t| + \textstyle\sum_{s=0}^{t-1} |Q'_s|) + |Q_t| + |K^t| \\ & + \textstyle\sum_{s=1}^{t} |Q'_s| + \sum_{s=1}^{t} |K^s||H| \\ =\ & N(|Q_t| + \textstyle\sum_{s=0}^{t-1} |Q'_s|) + |K^t| + |H| \sum_{s=1}^{t} |K^s|. \end{aligned} \tag{5.32}$$

Again from Equations (5.26) and (5.27), substitute Q_s and Q'_s by H, G and K, we get

$$\begin{aligned} & N(|Q_t| + \sum_{s=0}^{t-1} |Q'_s|) + |K^t| + |H| \sum_{s=1}^{t} |K^s| \\ \leq & |G^t - K^t| + |H| \sum_{s=0}^{t-1} |G^s - K^s| + |K^t| + |H| \sum_{s=1}^{t} |K^s| \\ = & \kappa(t). \end{aligned} \tag{5.33}$$

Chaining Equations (5.31), (5.32) and (5.33), it follows that, for all t-adjacent p, p', $|\mathbf{x}(t) - \mathbf{x}'(t)| \leq \kappa(t)$. Thus the lemma follows. □

Remark 5.2. The upper bound on the sensitivity at time t, $\kappa(t)$ has two components:

(a) $|K^t| + |H| \sum_{s=1}^{t} |K^s|$ over-approximates the change in agent i's state ($\mathbf{x}_i$) if its own preference changes by at most unity at each time up to t, and

(b) $|G^s - K^s| + |H| \sum_{s=0}^{t-1} |G^s - K^s|$ over-approximates the sum of the changes in other agents' states given agent i's preference changes by at most unity up to t.

Remark 5.3. $\kappa(t)$ is independent to the number of agents (N). It only depends on matrix K, the coupling constant c and time t. K is specified by the individual's control function as in Equation (5.16), which assumes to be stable. The more stable matrix K is, the faster $|K^t|$ decays to 0. The coupling constant c quantifies the influence of the aggregate on each individual agent. The matrix $G = cI + K$ captures the combined dynamics under the influence of the environment and the dynamics of the individual agents. The weaker physical coupling is, the smaller $|G^t|$ is. Therefore, we conclude that, as the individual agent dynamics becomes more stable or the physical coupling between agents becomes weaker, the sensitivity of the system decreases.

Remark 5.4. The dependence of $\kappa(t)$ on time t changes based on the stability of the K and G matrices. If G is stable, $\kappa(t)$ converges to a constant as $t \to \infty$. Otherwise if G is unstable, $\kappa(t)$ grows exponentially with t.

Theorems 5.2 and 5.4 immediately suggest an observation map (r) which guarantees differential privacy of the distributed linear control system.

Corollary 5.5. *For any time bound T and privacy parameter $\varepsilon > 0$, for $M_t \triangleq \frac{T\kappa(t)}{\varepsilon}$ and $\omega_i(t)$ chosen as noise vector of length n drawn independently from the distribution $Lap(M_t)$, the resulting observation map makes the linear distributed control system ε-differentially private up to time T.*

Example 5.2. Now we can apply the strategy explained above to Example 5.1, where $K = \frac{1}{5}I$ is a 2 by 2 matrix. $G = (c + \frac{1}{5})I$ in this case. By Theorem 5.4, the sensitivity is

$$\begin{aligned} \Delta(t) &\leq \kappa(t) = G^t + (I - K) \textstyle\sum_{s=0}^{t-1} G^s, \\ &= \frac{4+20c}{20-25c} + \frac{16-45c}{20-25c} \left(c + \frac{1}{5}\right)^t. \end{aligned}$$

As we stated in Remark 5.3, the sensitivity is independent of N. If G is stable, that is $|c + \frac{1}{5}| \leq 1$, the sensitivity $\Delta(t)$ is bounded and converges to a constant as $t \to \infty$. Otherwise, if $|c + \frac{1}{5}| > 1$, $\kappa(t)$ diverges. We choose the noise to be $M_t = \frac{\kappa(t)T}{\varepsilon}$. By Corollary 5.5, the system guarantees ε-differential privacy up to time T.

5.6.2 Cost of Privacy in Linear Distributed Control

The observation map of Corollary 5.5 adds independently drawn Laplace noise to the state of agent i observation at time t from the distribution $Lap(M_t)$. The noise parameter M_t depends on the individual's dynamics rather than the number of agents. In this section, we discuss the cost of privacy for this mechanism (see, Definition 5.4) compared to a perfectly observable system using the same controller.

Theorem 5.6. *The cost of privacy of the ε-differentially private mechanism $\mathcal{M}$ of Corollary 5.5 is inversely proportional to the number of agents N and the squared privacy parameter ε^2. In addition, if matrix G is stable, it is proportional to T^3. Otherwise if G is unstable, the cost of privacy grows exponentially with T.*

Proof. Given the ε-differentially private mechanism $\mathcal{M}$, the perfectly observable system $\mathcal{M}'$ is obtained by setting the noise values to be 0. We denote by $\bar{\mathbf{x}}_i(t)$ the state of agent i for $\mathcal{M}'$ at time t. From Equation (5.22), by fixing $\omega_i(t) \equiv 0$, we get

$$\bar{\mathbf{x}}_i(t) = K^t p_i(0) + \sum_{s=1}^{t} K^{t-s}(I - K)p_i(s).$$

We define a $n \times nN$ matrix $\mathbf{B} \triangleq \frac{c}{N}[I, \ldots, I]$. Let $x_i(t)$ be agent i's state corresponding to some execution of $\mathcal{M}(p)$. Again from Equation (5.22), the state of an individual agent i is

$$\mathbf{x}_i(t) = \bar{\mathbf{x}}_i(t) - \sum_{s=0}^{t-1} K^{t-s-1}\mathbf{B}\omega(s).$$

The cost of the mechanism $\mathcal{M}$ can be written as

$$\begin{aligned} cost_{\mathcal{M},i}(T) &= \mathbb{E}\left[\sum_{t=1}^{T} |\mathbf{x}_i(t) - p_i(t)|_2^2\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{T} |\bar{\mathbf{x}}_i(t) - \sum_{s=0}^{t-1} K^{t-s-1}\mathbf{B}\omega(s) - p_i(t)|_2^2\right] \\ &= \sum_{t=1}^{T} \mathbb{E}[|\bar{\mathbf{x}}_i(t) - p_i(t)|_2^2 + |\sum_{s=0}^{t-1} K^{t-s-1}\mathbf{B}\omega(s)|_2^2 \\ &\quad -2(\bar{\mathbf{x}}_i(t) - p_i(t))^\top \sum_{s=0}^{t-1} K^{t-s-1}\mathbf{B}\omega(s)]. \end{aligned}$$

The first term on the right-hand side is the cost of the system with perfect observations, that is, $cost_{\mathcal{M}',i}(T)$. The last term on the right-hand side is the expectation of a linear combination of zero-mean noise terms, and therefore, equals 0. By Definition 5.4,

$$\begin{aligned} CoP(\varepsilon, \mathcal{M}, T) &= \sup_{p,i}[cost_{\mathcal{M}(p),i}(T) - cost_{\mathcal{M}'(p),i}(T)] \\ &= \textstyle\sum_{t=1}^{T} \mathbb{E}\left[|\sum_{s=0}^{t-1} K^{t-s-1}\mathbf{B}\omega(s)|_2^2\right]. \end{aligned} \tag{5.34}$$

In our Laplace mechanism, for different time steps s, τ, $\omega(s)$ and $\omega(\tau)$ are independent. Thus,

$$\mathbb{E}[\omega(s)^\top \omega(\tau)] = \mathbb{E}[\omega(s)]^\top \mathbb{E}[\omega(\tau)] = 0.$$

Then, the right-hand side of Equation (5.34) reduces to

$$\sum_{t=1}^{T} \mathbb{E}\left[\sum_{s=0}^{t-1} \omega(s)^\top \mathbf{B}^\top (K^{t-s-1})^\top K^{t-s-1}\mathbf{B}\omega(s)\right].$$

Recall that each $\omega(s)$ consists of a noise vector $\omega_i(s)$ for each agent $i \in [N]$, and each of these vectors have n independent and identically distributed noise values drawn from $Lap(M_s)$. Each pair of vectors in $\omega(s)$ are independent. Denote $\omega^{(k)}(s)$, $k \in [nN]$, be the k-th element of the vector $\omega(s)$.

It follows that (a) for $k \neq j \in [nN]$, $\mathbb{E}\left[\omega^{(k)}(s)\omega^{(j)}(s)\right] = 0$, and (b) for any $k \in [nN]$, $\mathbb{E}\left[\omega^{(k)}(s)\omega^{(k)}(s)\right] = 2M_s^2$. Thus, the above expression is reduced

to

$$\sum_{t=1}^{T}\sum_{s=0}^{t-1} 2M_s^2 \mathrm{Tr}(\mathbf{B}^\top (K^{t-s-1})^\top K^{t-s-1}\mathbf{B}), \tag{5.35}$$

where $\mathrm{Tr}(A)$ stands for the trace of matrix A. Recall that $\mathbf{B} \triangleq \frac{c}{N}[I,\ldots,I]$. It follows that

$$\begin{aligned}&\mathrm{Tr}(\mathbf{B}^\top (K^{t-s-1})^\top K^{t-s-1}\mathbf{B})\\ =&\frac{c^2}{N}\mathrm{Tr}((K^{t-s-1})^\top K^{t-s-1}) = \frac{c^2}{N}|K^{t-s-1}|_2^2.\end{aligned}$$

Substituting the above equation into Equation (5.35) yields

$$CoP(\varepsilon,\mathcal{M},T) = \frac{2c^2}{N}\sum_{t=1}^{T}\sum_{s=0}^{t-1} M_s^2|K^{t-s-1}|_2^2.$$

By interchanging the order of summation we get

$$\begin{aligned}CoP(\varepsilon,\mathcal{M},T) =&\frac{2c^2}{N}\sum_{s=0}^{T-1}\sum_{t=s+1}^{T} M_s^2|K^{t-s-1}|_2^2\\ =&\frac{2c^2}{N}\sum_{s=0}^{T-1} M_s^2\sum_{t=0}^{T-s-1}|K^t|_2^2.\end{aligned} \tag{5.36}$$

Recall that in Corollary 5.5, $M_s = \frac{T\kappa(s)}{\varepsilon}$. Combining this with Equation (5.36), we have

$$CoP(\varepsilon,\mathcal{M},T) = \frac{2c^2T^2}{N\varepsilon^2}\sum_{s=0}^{T-1}\kappa(s)^2\sum_{t=0}^{T-s-1}|K^t|_2^2.$$

From the above expression it is clear $CoP(\varepsilon,\mathcal{M},T)$ is inversely proportional to N and ε^2. As the matrix K is stable, $\sum_{t=0}^{T-s-1}|K^t|_2^2$ converges to some constant as $T\to\infty$. By Remark 5.4, if G is stable then $\kappa(s)$ converges to some constant as $s\to\infty$, $\sum_{s=0}^{T-1}\kappa(s)^2$ grows linearly with T and we have $CoP(\varepsilon,\mathcal{M},T)\sim O(T^3)$. Otherwise if G is unstable, $\kappa(s)$ grows exponentially with s and $CoP(\varepsilon,\mathcal{M},T)$ grows exponentially with T. □

Example 5.3. Continuing with the system described in Example 5.1, we now establish the cost of privacy associated with the communication strategy of Equation (5.36). In this example, $K = 0.2I$. We choose the coupling parameter c to be 0.4. Then, the close-loop system is stable. Therefore, the sensitivity is bounded by $\kappa(t) = 1.2 - 0.2\times 0.6^t$. The cost of privacy of the

system with N agents at time T follows $\frac{0.24T^3}{N\varepsilon^2} + O(\frac{T^2}{N\varepsilon^2})$.

Example 5.4. We conclude with a simulation-based analysis of the traffic control Example 5.1. Consider a linear distributed control system in which each agent is a point on the plane moving toward a randomly chosen destination with dynamics described in Example 5.3 and control strategies given in Example 5.3. The cost of each agent is defined by the distance between its position to its destination. The coupling between agents is the repulsive force in the direction of the center of mass (CM) of the population. Thus, if the control of an individual fights the force too strongly without the knowledge of the CM then a higher cost is incurred. We numerically simulated the system with different levels of privacy and different distributions of destinations and make the following observations.

Figure 5.2 shows the relative costs of control with (blue) no communication and (green) private communication, with respect to cost of control with complete (or broadcast) communication. First of all, if both the initial positions and the destinations are chosen with 0 mean, then the CM of the population hovers around the origin and in that case, the contribution of the coupling is small. As a result, there is not much to be gained through communication and we see (Figure 5.2) that the cost of the system with privacy is comparable to the cost of the system with no communication. When the destination comes from some distributions slightly biased from 0, we start to see that the cost of control with private communication starts to become smaller compared to those of systems with no communications.

Figure 5.3 shows that for the same distribution of initial positions and destinations the cost of privacy changes as predicted by Theorem 5.6. First of all, a higher level of privacy comes with a higher cost (Figure 5.3a). As ε changes from 0.2 to 2, the CoP changes from 10 to 0.1. Secondly, larger number of agents (N) gives lower cost of privacy (Figure 5.3b). As N changes from 10 to 100, the CoP decreases from 4 to 0.4. And finally a longer time horizon (T) translates to higher costs (Figure 5.3c). The simulation results suggest that the cost of privacy roughly has the order of $O(\frac{T^3}{N\varepsilon^2})$.

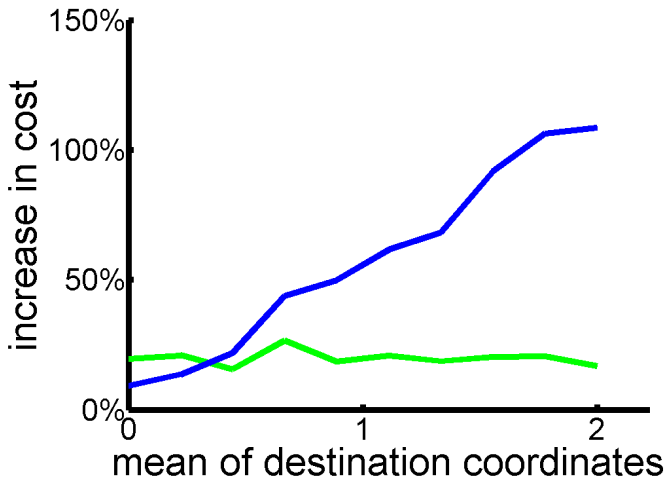
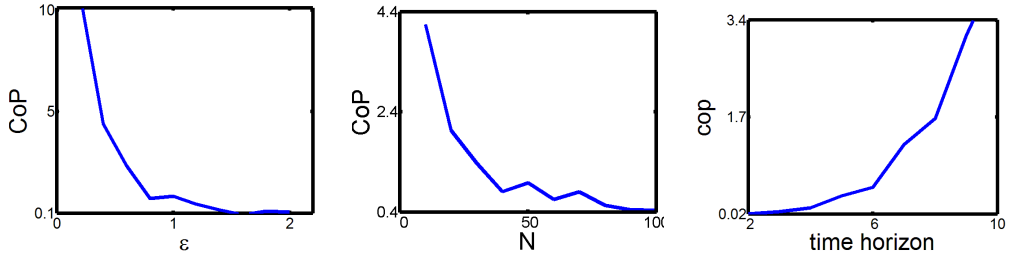


Figure 5.2: Increase in cost with biased sampled destinations. The blue and green lines capture the relative cost of control with no communication and private communication with respect to the cost of control with broadcast preferences, respectively.



(a) Cost of privacy vs. privacy level ε.

(b) Cost of privacy vs. number of agents N.

(c) Cost of privacy vs. time horizon T.

Figure 5.3: Cost of privacy for different privacy level, number of agents and time horizon.

5.7 Summary

We presented a general framework for studying cost of differential privacy for distributed control systems. We proposed a communication strategy by which individual agents can share noisy information about their states which preserves ε-differential privacy while aiding the estimation of the aggregate environment and therefore improving control performance. The distribution of the noise depends on the sensitivity with respect to the individual's data. Specializing to linear systems with quadratic costs, we showed that the sensitivity and therefore the standard deviation of the required noise is inde-

pendent of the number of participating agents. The sensitivity also decreases with the stability of the dynamics and with the weakening the environment's influence on an individual. For stable controllers, for preserving privacy over indefinite time horizons, the variance of the noise to be added is also independent of time. For unstable dynamics, on the other hand, the sensitivity can grow exponentially with time. The cost of ε-differential privacy for the proposed communication strategy up to time T for a system with N agents is at most $O(\frac{T^3}{N\varepsilon^2})$ for stable systems. This suggests that the proposed communication strategy is best suited for distributed control systems with many short-lived participants.

The proposed framework should enable us to study more sophisticated communication strategies that incur smaller costs for more persistent agents. Another direction for future research will be to establish lower bounds on the best cost of privacy that can be achieved through any communication strategy, not just the form proposed here.

Chapter 6

CONCLUSION

The objective of this thesis is to develop techniques that provably guarantee safety and privacy for cyber-physical systems (CPS). For different formalisms of CPS, we compute the bound on sensitivity of the system with respect to different parameters and use them in two ways. For invariance verification, bounds on sensitivity are used to generalize a single trajectory to get a tube that contains all neighboring trajectories. We show that checking these tubes suffices to prove invariance for the system. For privacy preservation, we compute sensitivity with respect to user data. Bounds on sensitivity are used to decide the noise distribution that obscures the exact user data. We will briefly summarize our main results and discuss possible future directions.

6.1 Summary of Contributions

In this thesis, we present three techniques for invariance verification and privacy preservation. Figure 6.1 illustrates the main concepts we introduced in this thesis and how they are connected. There are three streams of technical contributions.

In Chapter 3, we present a verification algorithm to prove invariance properties for networked dynamical networks with delayed interconnection. We propose the notion of IS discrepancy functions for subsystems which bounds the distance between trajectories with respect to the initial states and inputs. With IS discrepancy of subsystems, our approach syntactically constructs a reduced model of the whole network. The trajectory of the reduced model bounds the distance between trajectories of the entire network. Using the above results, we present an algorithm to over-approximate reach sets for networked dynamical systems. We further show that the over-approximations can be made arbitrarily precise. Therefore our verification algorithm is sound

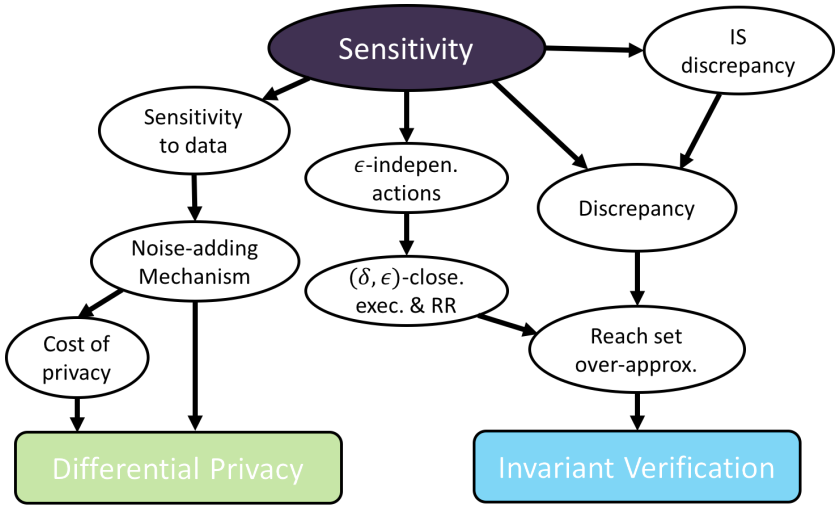


Figure 6.1: Concept graph of the thesis.

and relatively complete.

In Chapter 4, we develop a partial order reduction technique for infinite-state labeled transition systems. We propose the notion of ε-independent actions, such that the resulting states are within ε distance after executing a pair of ε-independent actions in any order. We define two executions to be (δ, ε)-close if their initial states are within δ distance and their action sequences are identical modulo swapping ε-independent actions. For any execution ξ, we present an algorithm for upper-bounding the distance between ξ and its (δ, ε)-close executions, namely the representative radius. With this algorithm, we can precisely over-approximate reach set of all (δ, ε)-close executions by examining one representative, which may lead to an exponential reduction in the number of paths that need to be explored for verifying a concurrent cyber-physical system.

In Chapter 5, we present a communication strategy that preserves differential privacy of participating agents. Central to this technique is the concept of sensitivity with respect to the private data. We show that, if each agent adds carefully designed noise—with a distribution depending on the sensitivity—to its communication, the ε-differential privacy of agents is protected. We present a framework for studying the privacy-performance trade-offs. We show that, for linear systems with stable dynamics, the cost of ε-differential privacy for the proposed communication strategy up to time T for a system with N agents is at most $O(\frac{T^3}{N\varepsilon^2})$. Hence, the cost decays to 0 as the number of participants increases.

6.2 Future Directions

Broadly speaking, the research on sensitivity-based verification of cyber-physical systems is relatively advanced. Several available tools implement the algorithms using different strategies and software libraries. These tools have been used to demonstrate applicability of these techniques on some realistic benchmark problems. The research on privacy of distributed cyber-physical systems is still in its early stages. There is considerable ongoing work on exploring different definitions for privacy and formulations for privacy preserving mechanisms.

6.2.1 Locally Independent Actions

In Chapter 4, we presented a reach set over-approximation methods exploiting the approximate commutativity property of ε-independent actions. There is a trade-off in efficiency and precision. Choosing large approximation parameter ε increase the number of ε-independent action pairs and expands the equivalent classes of action sequences. This way, our partial order reduction method is more efficient. However, a large ε leads to over-conservative over-approximation of the reach set.

In Definition 4.3, ε-independent actions are required to be approximately commutative globally. That is, from *any* state q, the resulting states after executing ε-independent actions in any order are within ε distance. The current definition ignores those action pairs that are approximately commutative *only* locally over parts of the state space. Hence, to make such a pair of actions ε-independent, the current method can lead to an over-conservative choice of the approximation parameter ε.

As a natural next step, the notion of approximate independent actions could be relaxed to actions that approximately commute locally. One possible technique involves splitting locally independent actions. Consider for example a pair of actions a and b that are only ε-independent over a subset of states S but nowhere else. If we split actions a and b respectively into two pairs of actions (a_0, a_1) and (b_0, b_1) such that actions a_0, b_0 are enabled in S and a_1, b_1 are enabled otherwise, then the new actions a_0 and b_0 are ε-independent. With techniques like this, we could choose tighter parameter ε and still preserve approximate independency for actions in some part of

state space. Hence, the precision of reach set computation could be improved without any loss of efficiency.

6.2.2 Verification of Hybrid Automata

In this thesis, we presented sensitivity-based invariant verification techniques for both discrete and continuous models of CPS. A natural next step is to extend these techniques to hybrid automata [10, 11, 12]. Hybrid automata can be viewed as a combination of transition systems and differential equations, where the states can update through either continuous evolution or discrete transitions.

In Chapter 3, we introduced an invariant verification algorithm for dynamical systems using IS discrepancy. To extend this approach to hybrid automata, one need techniques to handle discrete transitions. In [122, 52], the authors presented routines to detect possible discrete transitions and over-approximate states after transitions. Combing this technique with our IS discrepancy-based reachability analysis, one could possibly develop an invariant verification method for hybrid automata.

In Chapter 4, we presented a partial order reduction method for labeled transition systems. To extend the technique to hybrid automata, a straightforward approach is through abstraction. In model checking hybrid automata, it is standard to construct discrete abstraction [57], where our partial order reduction method can directly apply. However, accurate abstraction often requires excessive state space partitioning preventing this approach scaling to large models. For partial order reduction, abstraction is only required in some parts of the state space where commutative discrete transitions can occur. Hence, one interesting direction is to develop efficient abstraction technique for partial order reduction.

6.2.3 Differential Privacy with Nonlinear Dynamics

In Theorem 5.2, we presented a general sufficient condition for a differentially private mechanism in terms of the sensitivity to user data. With this sufficient condition, we developed a differentially private communication strategy for linear systems. A direct next step is to extend this result to systems with

nonlinear and hybrid dynamics. The main challenge in this direction is to compute sensitivity for these systems. There is no general techniques for computing the exact sensitivity. Several existing techniques estimate sensitivity by sampling [123, 66], however, they do not provide provable guarantees. More recently, techniques are developed to soundly over-approximate sensitivity to initial states [124, 48], however, whether they can be used for differential privacy remains to be explored. Furthermore, for many systems where user privacy is a major concern, exact global states are sometimes inaccessible. Hence, an interesting direction is to compute global sensitivity using only local information.

6.2.4 Tools and Applications

In the past few years, several end-to-end verification tools based on sensitivity (or discrepancy) analysis are developed, such as C2E2 [52], Breach [27], and Strong [125]. The main challenge for these tools is to compute sensitivity of large models efficiently. We have presented two techniques for computing sensitivity in Chapters 3 and 4. For our techniques to make a greater impact, future effort should be on building a robust implementation and connecting with existing verification tools.

Another direction is to apply our techniques to verify other examples of networked cyber-physical systems. Biological systems are naturally compositional, making them suitable examples for our invariant verification techniques. Examples of this type include pacemaker-heart interfaces [33, 126], surgical robots [127, 128], and neural systems [129, 130]. Our techniques could also aid verifying distributed and swarm robots, in applications such as flocking [131], connection maintenance [132], and collision avoidance [133, 134]. Power networks also need high assurance of safety [135, 136]. Some of these systems evolve according to nonlinear differential algebraic equations, which require theoretical developments for sensitivity analysis.

REFERENCES

[1] J. Machowski, J. Bialek, and J. Bumby, *Power System Dynamics: Stability and Control*. John Wiley & Sons, 2011.

[2] R. B. Perry, M. M. Madden, W. Torres-Pomales, and R. W. Butler, *The Simplified Aircraft-Based Paired Approach with the ALAS Alerting Algorithm*. NASA/TM-2013-217804, 2013.

[3] P. S. Duggirala, L. Wang, S. Mitra, M. Viswanathan, and C. Munoz, "Temporal precedence checking for switched models and its application to a parallel landing protocol," *Proceedings of International Symposium on Formal Methods*. Springer, 2014, pp. 215–229.

[4] T. T. Johnson, Z. Hong, and A. Kapoor, "Design verification methods for switching power converters," *Proceedings of Power and Energy Conference at Illinois (PECI), 2012 IEEE*. IEEE, 2012, pp. 1–6.

[5] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[6] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno et al., "Comprehensive experimental analyses of automotive attack surfaces," *Proceedings of USENIX Security Symposium*. San Francisco, 2011.

[7] Y. Mo and B. Sinopoli, "Secure control against replay attacks," *Communication, Control, and Computing, Proceedings of 47th Annual Allerton Conference on*. IEEE, 2009, pp. 911–918.

[8] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," *Proceedings of 2008 IEEE Symposium on Security and Privacy*. IEEE, 2008, pp. 111–125.

[9] T. Jeske, "Floating car data from smartphones: What Google and Waze know about you and how hackers can control traffic," *Proceedings of the BlackHat Europe*, pp. 1–12, 2013.

[10] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," *Hybrid Systems*. Springer, 1993, pp. 209–229.

[11] S. Mitra, "A verification framework for hybrid systems," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[12] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager, *The Theory of Timed I/O Automata*. Morgan Claypool, November 2005, also available as Technical Report MIT-LCS-TR-917.

[13] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*. Springer Science & Business Media, 2010.

[14] R. David and H. Alla, "On hybrid petri nets," *Discrete Event Dynamic Systems*, vol. 11, no. 1-2, pp. 9–40, 2001.

[15] D. Liberzon, *Switching in Systems and Control*. Springer Science & Business Media, 2012.

[16] A. Platzer, "Differential dynamic logic for hybrid systems," *Journal of Automated Reasoning*, vol. 41, no. 2, pp. 143–189, 2008.

[17] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[18] P. J. L. Cuijpers and M. A. Reniers, "Hybrid process algebra," *The Journal of Logic and Algebraic Programming*, vol. 62, no. 2, pp. 191–245, 2005.

[19] G. Filatrella, A. H. Nielsen, and N. F. Pedersen, "Analysis of a power grid using a kuramoto-like model," *The European Physical Journal B*, vol. 61, no. 4, pp. 485–491, 2008.

[20] L. Chaimowicz, V. Kumar, and M. F. Campos, "A paradigm for dynamic coordination of multiple robots," *Autonomous Robots*, vol. 17, no. 1, pp. 7–21, 2004.

[21] R. Grosu, G. Batt, F. H. Fenton, J. Glimm, C. Le Guernic, S. A. Smolka, and E. Bartocci, "From cardiac cells to genetic regulatory networks," *Computer Aided Verification*. Springer, 2011, pp. 396–411.

[22] M. Sipser, *Introduction to the Theory of Computation*. Thomson Course Technology Boston, 2006, vol. 2.

[23] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 1999.

[24] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. MIT Press, 2008.

[25] R. Cukier, H. Levine, and K. Shuler, "Nonlinear sensitivity analysis of multiparameter model systems," *Journal of Computational Physics*, vol. 26, no. 1, pp. 1–42, 1978.

[26] T. Maly and L. R. Petzold, "Numerical methods and software for sensitivity analysis of differential-algebraic systems," *Applied Numerical Mathematics*, vol. 20, no. 1, pp. 57–79, 1996.

[27] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," *Computer Aided Verification*. Springer, 2010, pp. 167–170.

[28] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, *S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems*. Springer, 2011.

[29] P. Duggirala, S. Mitra, and M. Viswanathan, "Verification of annotated models from executions," *International Conference on Embedded Software*, 2013.

[30] Z. Huang and S. Mitra, "Proofs from simulations and modular annotations," *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*. ACM, 2014, pp. 183–192.

[31] Z. Huang, C. Fan, A. Mereacre, S. Mitra, and M. Kwiatkowska, "Invariant verification of nonlinear hybrid automata networks of cardiac cells," *Computer Aided Verification*. Springer, 2014, pp. 373–390.

[32] U. Topcu, A. Packard, and R. Murray, "Compositional stability analysis based on dual decomposition," *Decision and Control, Proceedings of the 48th IEEE Conference on*, Dec 2009, pp. 1175–1180.

[33] Z. Huang, C. Fan, A. Mereacre, S. Mitra, and M. Kwiatkowska, "Simulation-based verification of cardiac pacemakers with guaranteed coverage," *IEEE Design & Test*, vol. 32, no. 5, pp. 27–34, 2015.

[34] P. Godefroid, J. van Leeuwen, J. Hartmanis, G. Goos, and P. Wolper, *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*. Springer Heidelberg, 1996, vol. 1032.

[35] D. Peled, "Ten years of partial order reduction," in *International Conference on Computer Aided Verification*. Springer, 1998, pp. 17–28.

[36] J. Herrera, D. Work, R. Herring, X. Ban, Q. Jacobson, and A. Bayen, "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment," *Transportation Research Part C*, vol. 18, no. 4, pp. 568–583, August 2010.

[37] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013, pp. 901–914.

[38] M. Xue, W. Wang, and S. Roy, "Security concepts for the dynamics of autonomous vehicle networks," *Automatica*, vol. 50, no. 3, pp. 852–857, 2014.

[39] F. Koufogiannis, S. Han, and G. J. Pappas, "Computation of privacy-preserving prices in smart grids," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 2142–2147.

[40] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data." in *19th Annual Network & Distributed System Security Symposium (NDSS)*, vol. 2, no. 3, 2011, p. 4.

[41] Y. Zhang, Y. Zhang, and K. Ren, "Distributed privacy-preserving access control in sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 8, pp. 1427–1438, Aug 2012.

[42] C. Dwork, "Differential privacy," *Automata, Languages and Programming*. Springer, 2006, pp. 1–12.

[43] C. Dwork, "Differential privacy: A survey of results," *Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[44] C. Dwork, M. Naor, G. Rothblum, and T. Pitassi, "Differential privacy under continual observation," *Proceedings of the 42nd ACM Symposium on Theory of Computing*, 2010.

[45] M. Hardt and K. Talwar, "On the geometry of differential privacy," *Proceedings of the 42nd ACM Symposium on Theory of Computing*. ACM, 2010, pp. 705–714.

[46] Z. Huang, S. Mitra, and G. Dullerud, "Differentially private iterative synchronous consensus," *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, New York, NY, USA: ACM, 2012. pp. 81–90.

[47] J. Le Ny and G. J. Pappas, "Differentially private filtering," *Automatic Control, IEEE Transactions on*, vol. 59, no. 2, pp. 341–354, 2014.

[48] C. Fan and S. Mitra, "Bounded verification with on-the-fly discrepancy computation," in *International Symposium on Automated Technology for Verification and Analysis (ATVA), 2015*, 2015.

[49] M. Jha and S. Raskhodnikova, "Testing and reconstruction of Lipschitz functions with applications to data privacy," *SIAM Journal on Computing*, vol. 42, no. 2, pp. 700–731, 2013.

[50] D. Angeli, "A Lyapunov approach to incremental stability properties," *Automatic Control, IEEE Transactions on*, vol. 47, no. 3, pp. 410–421, 2002.

[51] E. M. Aylward, P. A. Parrilo, and J.-J. E. Slotine, "Stability and robustness analysis of nonlinear systems via contraction metrics and SoS programming," *Automatica*, vol. 44, no. 8, pp. 2163–2170, 2008.

[52] P. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2E2: A verification tool for stateflow models," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, vol. 9035. Springer Berlin Heidelberg, 2015, pp. 68–82.

[53] T. A. Henzinger, *The Theory of Hybrid Automata*. Springer, 2000.

[54] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995.

[55] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*. ACM, 1995, pp. 373–382.

[56] E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Informatica*, vol. 43, no. 7, pp. 451–476, 2007.

[57] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[58] E. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald, "Abstraction and counterexample-guided refinement in model checking of hybrid systems," *International Journal of Foundations of Computer Science*, vol. 14, no. 04, pp. 583–604, 2003.

[59] X. Chen, E. Abrahám, and S. Sankaranarayanan, "Taylor model flowpipe construction for non-linear hybrid systems," in *Real-Time Systems Symposium, 2012 IEEE 33rd*. IEEE, 2012, pp. 183–192.

[60] A. Donzé and O. Maler, "Systematic simulation using sensitivity analysis," in *Hybrid Systems: Computation and Control*. Springer, 2007, pp. 174–189.

[61] Z. Huang, C. Fan, and S. Mitra, "Bounded invariant verification for time-delayed nonlinear networked dynamical systems," *Nonlinear Analysis: Hybrid Systems*, 2016.

[62] L. Zou, M. Fränzle, N. Zhan, and P. N. Mosaad, "Automatic verification of stability and safety for delay differential equations," in *Computer Aided Verification*. Springer, 2015, pp. 338–355.

[63] J. Nagumo, S. Arimoto, and S. Yoshizawa, "An active pulse transmission line simulating nerve axon," *Proceedings of the IRE*, vol. 50, no. 10, pp. 2061–2070, 1962.

[64] Y. Kuang, *Delay Differential Equations: With Applications in Population Dynamics*. Academic Press, 1993.

[65] D. Angeli, "Further results on incremental input-to-state stability," *Automatic Control, IEEE Transactions on*, vol. 54, no. 6, pp. 1386–1391, 2009.

[66] G. Wood and B. Zhang, "Estimation of the Lipschitz constant of a function," *Journal of Global Optimization*, vol. 8, no. 1, pp. 91–103, 1996.

[67] P. Benner, J.-R. Li, and T. Penzl, "Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems," *Numerical Linear Algebra with Applications*, vol. 15, no. 9, pp. 755–777, 2008.

[68] E. D. Sontag, "Comments on integral variants of ISS," *Systems & Control Letters*, vol. 34, no. 1-2, pp. 93 – 100, 1998.

[69] D. Angeli, E. D. Sontag, and Y. Wang, "A characterization of integral input-to-state stability," *Automatic Control, IEEE Transactions on*, vol. 45, no. 6, pp. 1082–1097, 2000.

[70] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 1992.

[71] P. S. Duggirala, C. Fan, S. Mitra, and M. Viswanathan, "Meeting a powertrain verification challenge," *In Proceedings of International Conference on Computer Aided Verification (CAV 2015)*, 2015.

[72] CAPD, "Computer assisted proofs in dynamics," 2002. [Online]. Available: http://www.capd.ii.uj.edu.pl/

[73] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss, "Validated solutions of initial value problems for ordinary differential equations," *Applied Mathematics and Computation*, vol. 105, no. 1, pp. 21–68, 1999.

[74] O. Bouissou and M. Martel, "GRKLib: A guaranteed Runge Kutta library," in *Scientific Computing, Computer Arithmetic and Validated Numerics, 2006. SCAN 2006. 12th GAMM-IMACS International Symposium on*. IEEE, 2006, pp. 8–8.

[75] A. Bellen and M. Zennaro, *Numerical Methods for Delay Differential Equations*. Oxford University Press, 2013.

[76] L. F. Shampine and S. Thompson, "Solving DDEs in Matlab," *Applied Numerical Mathematics*, vol. 37, no. 4, pp. 441–458, 2001.

[77] K. Engelborghs, T. Luzyanina, and D. Roose, "Numerical bifurcation analysis of delay differential equations using DDE-BIFTOOL," *ACM Transactions on Mathematical Software (TOMS)*, vol. 28, no. 1, pp. 1–21, 2002.

[78] L. F. Shampine and S. Thompson, "Numerical solution of delay differential equations," in *Delay Differential Equations*. Springer, 2009, pp. 1–27.

[79] L. Scardovi and R. Sepulchre, "Synchronization in networks of identical linear systems," *Automatica*, vol. 45, no. 11, pp. 2557–2562, 2009.

[80] Z. Huang and S. Mitra, "Computing bounded reach sets from sampled simulation traces," *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*. ACM, 2012, pp. 291–294.

[81] A. Mazurkiewicz, "Concurrent program schemes and their interpretations," *DAIMI Report Series*, vol. 6, no. 78, 1977.

[82] R. Alur, R. K. Brayton, T. A. Henzinger, S. Qadeer, and S. K. Rajamani, "Partial-order reduction in symbolic state space exploration," in *International Conference on Computer Aided Verification*. Springer, 1997, pp. 340–351.

[83] C. Flanagan and P. Godefroid, "Dynamic partial-order reduction for model checking software," in *ACM Sigplan Notices*, vol. 40, no. 1. ACM, 2005, pp. 110–121.

[84] E. M. Clarke, O. Grumberg, M. Minea, and D. Peled, "State space reduction using partial order techniques," *International Journal on Software Tools for Technology Transfer*, vol. 2, no. 3, pp. 279–287, 1999.

[85] E. Clarke, S. Jha, and W. Marrero, "Partial order reductions for security protocol verification," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2000, pp. 503–518.

[86] C. Baier, M. Größer, and F. Ciesinski, "Partial order reduction for probabilistic systems." in *QEST*, vol. 4, 2004, pp. 230–239.

[87] D. Peled, "Verification for robust specification," in *International Conference on Theorem Proving in Higher Order Logics*. Springer, 1997, pp. 231–241.

[88] Y. Yang, X. Chen, G. Gopalakrishnan, and R. M. Kirby, "Efficient stateful dynamic partial order reduction," in *International SPIN Workshop on Model Checking of Software*. Springer, 2008, pp. 288–305.

[89] P. Abdulla, S. Aronis, B. Jonsson, and K. Sagonas, "Optimal dynamic partial order reduction," in *ACM SIGPLAN Notices*, vol. 49, no. 1. ACM, 2014, pp. 373–384.

[90] R. Majumdar and I. Saha, "Symbolic robustness analysis," in *Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE*. IEEE, 2009, pp. 355–363.

[91] S. Chaudhuri, S. Gulwani, and R. Lublinerman, "Continuity and robustness of programs," *Communications of the ACM*, vol. 55, no. 8, pp. 107–115, 2012.

[92] R. Samanta, J. V. Deshmukh, and S. Chaudhuri, "Robustness analysis of networked systems," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 2013, pp. 229–247.

[93] V. Blondel, J. M. Hendrickx, A. Olshevsky, J. Tsitsiklis et al., "Convergence in multiagent coordination, consensus, and flocking," in *IEEE Conference on Decision and Control*, vol. 44, no. 3. IEEE; 1998, 2005, p. 2996.

[94] S. Mitra and K. M. Chandy, "A formalized theory for verifying stability and convergence of automata in PVS," in *International Conference on Theorem Proving in Higher Order Logics*. Springer, 2008, pp. 230–245.

[95] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[96] J. L. Welch and N. Lynch, "A new fault-tolerant algorithm for clock synchronization," *Information and Computation*, vol. 77, no. 1, pp. 1–36, 1988.

[97] I.-K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y.-C. Wu, "Clock synchronization in wireless sensor networks: An overview," *Sensors*, vol. 9, no. 1, pp. 56–85, 2009.

[98] A. Fehnker and F. Ivančić, "Benchmarks for hybrid systems verification," *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2004, pp. 326–341.

[99] D. Mitra, "An asynchronous distributed algorithm for power control in cellular radio systems," *Wireless and Mobile Communications*. Springer, 1994, pp. 177–186.

[100] L. Fang and P. J. Antsaklis, "Information consensus of asynchronous discrete-time multi-agent systems," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 1883–1888.

[101] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *Proceedings of TCC*, 2006.

[102] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Foundations of Computer Science, 2007. FOCS '07. 48th Annual IEEE Symposium on*, oct. 2007, pp. 94 –103.

[103] A. Friedman and A. Schuster, "Data mining with differential privacy," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2010, pp. 493–502.

[104] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Advances in Neural Information Processing Systems*, 2009, pp. 289–296.

[105] J. Le Ny and G. J. Pappas, "Differentially private Kalman filtering," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on.* IEEE, 2012, pp. 1618–1625.

[106] M. Hale and M. Egerstedt, "Cloud-based optimization: A quasi-decentralized approach to multi-agent coordination," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, Dec 2014, pp. 6635–6640.

[107] M. Hale and M. Egerstedty, "Differentially private cloud-based multi-agent optimization with constraints," in *American Control Conference (ACC), 2015*, July 2015, pp. 1235–1240.

[108] S. Han, U. Topcu, and G. Pappas, "Differentially private convex optimization with piecewise affine objectives," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, Dec 2014, pp. 2160–2166.

[109] Y. Mo and R. Murray, "Privacy preserving average consensus," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, Dec 2014, pp. 2154–2159.

[110] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private average consensus: Obstructions, trade-offs, and optimal algorithm design," *ArXiv Preprint ArXiv:1512.09039*, 2015.

[111] J. Le Ny, "On differentially private filtering for event streams," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE, 2013, pp. 3481–3486.

[112] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, and C. Palamidessi, "Broadening the scope of differential privacy using metrics," in *Privacy Enhancing Technologies*. Springer, 2013, pp. 82–102.

[113] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor, "Optimizing linear counting queries under differential privacy," in *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '10. New York, NY, USA: ACM, 2010. pp. 123–134.

[114] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Advances in Cryptology - EUROCRYPT 2006*, ser. Lecture Notes in Computer Science, S. Vaudenay, Ed. Springer Berlin Heidelberg, 2006, vol. 4004, pp. 486–503.

[115] Q. Geng and P. Viswanath, "Optimal noise-adding mechanism in differential privacy," *CoRR*, vol. abs/1212.1186, 2012.

[116] M. Hardt and K. Talwar, "On the geometry of differential privacy," in *Proceedings of the 42nd ACM Symposium on Theory of Computing*, ser. STOC '10. New York, NY, USA: ACM, 2010. pp. 705–714.

[117] J. Reed and B. C. Pierce, "Distance makes the types grow stronger: a calculus for differential privacy," in *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*, ser. ICFP '10. New York, NY, USA: ACM, 2010. pp. 157–168.

[118] Y. Wang, Z. Huang, S. Mitra, and G. Dullerud, "Entropy-minimizing mechanism for differential privacy of discrete-time linear feedback systems," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, Dec 2014, pp. 2130–2135.

[119] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking*. ACM, 2015, p. 4.

[120] J. Le Ny and G. Pappas, "Differentially private filtering," *Automatic Control, IEEE Transactions on*, vol. 59, no. 2, pp. 341–354, Feb 2014.

[121] J. Le Ny, A. Touati, and G. J. Pappas, "Real-time privacy-preserving model-based estimation of traffic flows," in *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems*. IEEE Computer Society, 2014, pp. 92–102.

[122] P. S. Duggirala, "Dynamic analysis of cyber-physical systems," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2015.

[123] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth sensitivity and sampling in private data analysis," in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*. ACM, 2007, pp. 75–84.

[124] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala, "Automatic reachability analysis for nonlinear hybrid models with C2E2," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 531–538.

[125] Y. Deng, A. Rajhans, and A. A. Julius, "STRONG: A trajectory-based verification toolbox for hybrid systems," in *International Conference on Quantitative Evaluation of Systems*. Springer, 2013, pp. 165–168.

[126] Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam, "Modeling and verification of a dual chamber implantable pacemaker," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2012, pp. 188–203.

[127] S. Kumar, P. Singhal, and V. N. Krovi, "Computer-vision-based decision support in surgical robotics," *IEEE Design & Test*, vol. 32, no. 5, pp. 89–97, 2015.

[128] K. Miller, "Experimental verification of modeling of delta robot dynamics by direct application of Hamilton's principle," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 1. IEEE, 1995, pp. 532–537.

[129] Y. P. Gad and T. J. Anastasio, "Simulating the shaping of the fastigial deep nuclear saccade command by cerebellar Purkinje cells," *Neural Networks*, vol. 23, no. 7, pp. 789–804, 2010.

[130] P. Dean and J. Porrill, "Evaluating the adaptive-filter model of the cerebellum," *The Journal of Physiology*, vol. 589, no. 14, pp. 3459–3470, 2011.

[131] T. T. Johnson and S. Mitra, "Safe flocking in spite of actuator faults using directional failure detectors," *Journal of Nonlinear Systems and Applications*, vol. 73, p. 95, 2011.

[132] A. F. Winfield, W. Liu, J. Nembrini, and A. Martinoli, "Modelling a wireless connected swarm of mobile robots," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 241–266, 2008.

[133] P. S. Duggirala, T. T. Johnson, A. Zimmerman, and S. Mitra, "Static and dynamic analysis of timed distributed traces," in *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*. IEEE, 2012, pp. 173–182.

[134] S. Bak, Z. Huang, F. A. T. Abad, and M. Caccamo, "Safety and progress for distributed cyber-physical systems with unreliable communication," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 14, no. 4, p. 76, 2015.

[135] D. J. Hill and I. M. Mareels, "Stability theory for differential/algebraic systems with application to power systems," in *Robust Control of Linear Systems and Nonlinear Control*. Springer, 1990, pp. 437–445.

[136] P. W. Sauer and M. Pai, *Power System Dynamics and Stability*. John Wiley & Sons, 1997.