STATE ESTIMATION IN MULTI-AGENT DECISION AND CONTROL SYSTEMS

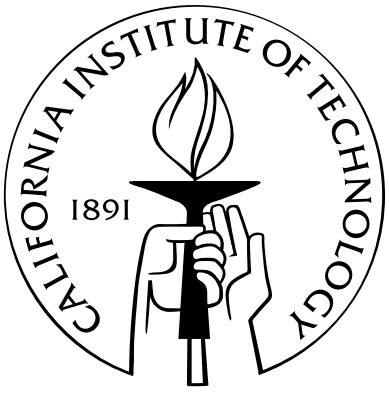
Thesis by

Domitilla Del Vecchio

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2005

(Defended March 29, 2005)

To my mother Daniela and to my dear Lorenzo

Acknowledgements

I would like to thank my committee members: Professors Richard M. Murray, Pietro Perona, Eric Klavins, John Doyle, and Jerry Marsden. In particular, I would like to thank Professors Pietro Perona and Eric Klavins for their substantial contributions to my research. A special thanks goes to my adviser, Professor Richard M. Murray, for being the best guide I could have ever had during the duration of my Ph.D. I wish to thank my family, Daniela, Raffaele, and Liana, for being always very close to me even if physically very far. I would like to deeply thank Lorenzo for having accompanied me all along the not-so-easy path that lead me to obtain my Ph.D. Finally, I would like to thank all of my friends, with whom I shared many, many things for all the duration of my studies.

Abstract

This thesis addresses the problem of estimating the state in multi-agent decision and control systems. In particular, a novel approach to state estimation is developed that uses partial order theory in order to overcome some of the severe computational complexity issues arising in multi-agent systems. Within this approach, state estimation algorithms are developed that enjoy provable convergence properties and are scalable with the number of agents.

The dynamic evolution of the systems under study are characterized by the interplay of continuous and discrete variables. Continuous variables usually represent physical quantities such as position, velocity, voltage, and current, while the discrete variables usually represent quantities internal to the decision protocol that are used for coordination, communication, and control. Within the proposed state estimation approach, the estimation of continuous and discrete variables is developed in the same mathematical framework as a joint continuous-discrete space is considered for the estimator. This way, the dichotomy between the continuous and discrete world is overcome for the purpose of state estimation.

Application examples are considered, which include the state estimation in competitive multi-robot systems and in multi-agent discrete event systems, and the monitoring of distributed environments.

Contents

List of Figures

Chapter 1

Introduction

Logic and decision making are playing increasingly large roles in modern control systems, and virtually all modern control systems are implemented using digital computers. Examples include aerospace systems, transportation systems (air, automotive, and rail), communication networks (wired, wireless, and cellular), and supply networks (electrical power and manufacturing). The evolution of these systems is determined by the interplay of continuous dynamics and logic. The continuous variables can represent quantities such as position, velocity, acceleration, voltage, current, etc., while the discrete variables can represent the state of the decision and communication protocol that is used for coordination and control. Most of these systems are also multi-agent, in which an agent can be, for example, a wireless device, a micro-controller, a robot, a piece of machinery, a piece of hardware or software, or even a human. The need for understanding and analyzing the behavior of these systems is compelling. However, the coupling of continuous dynamics and decision protocols renders the system under study interesting and complicated enough that new tools are needed for the sake of analysis and control. Also, multi-agent systems are usually affected by the combinatorial explosion of the state space that renders most of the existing state estimation algorithms inapplicable.

The problem of estimating the state of a decision and control system has been addressed by several authors for control or as a means for solving monitoring or surveillance problems in distributed environments. In the hybrid systems literature, Bemporad et al. [7] propose the notion of incremental observability for piecewise affine systems and construct a dead-beat observer that requires large amounts of computation. Balluchi et al. [4] combine a

location observer with a Luenberger observer to design hybrid observers that identify the location in a finite number of steps and converge exponentially to the continuous state. However, if the number of locations is large, as in the systems that we consider, such an approach is impracticable. In Balluchi et al., sufficient conditions for a linear hybrid system to be final state determinable are given [5]. In Alessandri et al., Luenberger-like observers are proposed for hybrid systems where the system location is known [2, 3]. Vidal et al. [43] derive sufficient and necessary conditions for observability of discrete time jump-linear systems, based on a simple rank test on the parameters of the model. In later work [44], these notions are generalized to the case of continuous time jump linear systems. For jump Markov linear systems, Costa and do Val derive a test for observability [18], and Cassandra et al. propose an approach to optimal control for partially observable Markov decision processes [15]. For continuous time hybrid systems, De Santis et al. propose a definition of observability based on the possibility of reconstructing the system state, and testable conditions for observability are provided [28].

In the discrete event literature, observability has been defined by Ramadge [38], for example, who derives a test for current state observability. Oishi et al. [37] derive a test for immediate observability in which the state of the system can be unambiguously reconstructed from the output associated with the current state and last and next events. Özveren et al. [22] and Caines [13, 14] propose discrete event observers based on the construction of the current-location observation tree that is impracticable when the number of locations is large, which is our case. Observability is also considered in the context of distributed monitoring and control in industrial automation, where agents are cooperating to perform system-level tasks such as failure detection and identification on the basis of local information [39]. Diaz et al. consider observers for formal on-line validation of distributed systems, in which the on-line behavior is checked against a formal model [29]. In the context of sensor networks, state estimation covers a fundamental role when solving surveillance and monitoring tasks in which the state usually has several components, such as the position of an agent, its identity, and its intent (see for example [17] or [11]).

The main contribution of this work is to design state estimators for decision and control systems that overcome severe complexity issues encountered in previous work ([4, 13, 14]).

These complexity issues render prohibitive the estimation problem for systems with a large discrete state space, which is often the case in multi-agent systems. Our point of view is that some of the complexity issues, such as those encountered in [13] or [4], can be avoided by finding a good way of representing the sets of interest and by finding a good way of computing maps on them. As a naive example, consider the set $\mathcal{S}$ of all natural numbers between one and one thousand. This set is usually represented as an interval in $\mathbb{N}$, that is $\mathcal{S} = [1, 1000]$, so that the listing of all the elements it contains is not necessary for representing it. Suppose we want to know what the set $\mathcal{S}$ is mapped to by a map ϕ that associates each element n with the element $n + 2$. Clearly, to compute $\phi(\mathcal{S})$ we do not need to compute ϕ on each element of $\mathcal{S}$ and collect all the results. In fact, it is easy to see that $\phi(\mathcal{S}) = [3, 1002]$, that is, we just compute ϕ on the least and maximum elements of $\mathcal{S}$ to obtain the least and maximum elements of $\phi(\mathcal{S})$, which are then used to represent the latter set. This simplification is possible thanks to the order structure naturally associated with $\mathbb{N}$ and thanks to the structure of the map ϕ.

These ideas are extended by using partial order theory to an arbitrary set, which might be more complicated than a set in $\mathbb{N}$ and might contain continuous components. Partial order theory has been used historically in theoretical computer science to prove properties about convergence of algorithms [19]. It has also been used for studying controllability properties of finite state machines [12] and for approaching the state explosion problem in the verification of concurrent systems [31]. In this work, we exploit partial order theory to estimate the state in systems with a large discrete space. In particular, given a system Σ defined on its space of variables, we extend it to a larger space of variables that has lattice structure so as to obtain an extended system $\tilde{\Sigma}$. Under certain properties verified by the extension $\tilde{\Sigma}$, an observer for system Σ can be constructed, which updates at each step only two variables. It updates the least and greatest element of the set of all values of variables compatible with the output sequence and with the dynamics of Σ. The structure of the obtained observer resembles the structure of the Luenberger observer or a Kalman filter as it is obtained by "copying" the dynamics of the system Σ and by correcting it according to the measured output values.

This work is concerned with the estimation of the discrete state in case the continuous

state is measured, and with the estimation of the whole system state in case a cascade structure of the estimator is possible. Within this context, the proposed estimation approach is also general as it applies to any observable system. In fact, we show that a system is observable if and only if there is a lattice in which the extended system satisfies the requirements for the construction of the proposed estimator. Within the state estimation framework that we develop, the estimation of the discrete and continuous part of the state space is handled in a unified way. In fact, there is no need to implement both a continuous state estimator that relies on classical control theory and a discrete state estimator based on automata theory, as done in most previous work. This is achieved by using a partial order to establish relationships between elements of a discrete space in analogy to how a metric establishes relationships between elements in a continuous space.

The contents of this work are organized as follows. In Chapter 2, some of the basic mathematical machinery on partial order theory and transition systems is introduced. Observability notions are introduced as well, and enumeration approaches to state estimation are reviewed. In Chapter 3, the state estimation problem is re-cast on a partial order and a solution is proposed for estimating the discrete state of a deterministic system when the continuous state is measured. Chapter 4 shows that the proposed approach applies to any observable system and thus is general. In Chapter 5, the results of Chapter 3 and of Chapter 4 are generalized to the case of nondeterministic systems. In Chapter 6, the results of the previous chapters are extended to the case of estimation of both discrete and continuous variables assuming a cascade for of the estimator. A multi-robot system involving two teams competing against each other is used through these chapters as a leading example. In Chapter 7, more application examples are proposed along with possible extensions.

Chapter 2

Basic Concepts

In this chapter, we review some basic notions that will be used throughout this work. First, we give some background on partial order and lattice theory in Section 2.1 (for more details the reader is referred to [21, 1]). The theory of partial orders, while standard in computer science, may be less well known to the intended audience of this thesis. Then, the class of systems under study is introduced in Section 2.2, i.e., deterministic transition systems, and the state estimation problem is defined. Finally, we show a solution to the problem in Section 2.3, an enumeration method that has been most often used in previous work.

2.1 Partial Order Theory

A partial order is a set χ with a partial order relation "$\leq$", and we denote it by the pair $(\chi, \leq)$. For any $x, w \in \chi$, $\sup\{x, w\}$ is the smallest element that is larger than both x and w. In a similar way, $\inf\{x, w\}$ is the largest element that is smaller than both x and w. We define the *join* "$\vee$" and the *meet* "$\wedge$" of two elements x and w in χ as

1. $x \vee w := \sup\{x, w\}$ and $x \wedge w := \inf\{x, w\}$;

2. if $S \subseteq \chi$, $\bigvee S := \sup\ S$, and $\bigwedge S := \inf\ S$.

Let $(\chi, \leq)$ be a partial order. If $x \wedge w \in \chi$ and $x \vee w \in \chi$ for any $x, w \in \chi$, then $(\chi, \leq)$ is a *lattice*. In Figure 2.1, we illustrate Hasse diagrams [21] showing partially ordered sets.

From the diagrams, it is easy to tell when one element is less than another: $x < w$ if and only if there is a sequence of connected line segments moving upward from x to w.

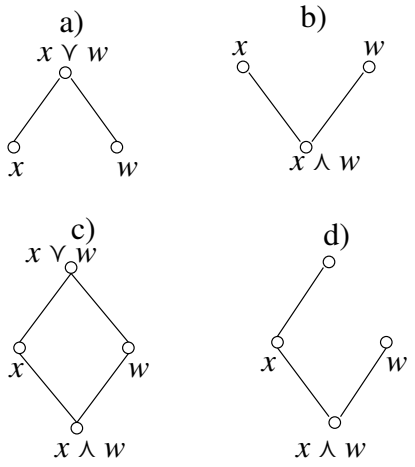


Figure 2.1: In diagram a) and b), x and w are not related, but they have a join and a meet, respectively. In diagram c), we show a complete lattice. In diagram d), we show a partially ordered set that is not a lattice, since the elements x and w have a meet, but not a join.

Let $(\chi, \leq)$ be a partial order. Then $(\chi, \leq)$ is a *chain* if for all $x, w \in \chi$, either $x \leq w$ or $w \leq x$, that is, any two elements are comparable. If instead any two elements are not comparable, i.e., $x \leq y$ if and only if $x = y$, $(\chi, \leq)$ is said to be an *anti-chain*. If $x < w$ and there is no other element in between x and w, we write $x \ll w$.

Let $(\chi, \leq)$ be a lattice and let $S \subseteq \chi$ be a non-empty subset of χ. Then, $(S, \leq)$ is a *sublattice* of χ if $a, b \in S$ implies that $a \vee b \in S$ and $a \wedge b \in S$. If any sublattice of χ contains its least and greatest elements, then $(\chi, \leq)$ is called *complete*. Any finite lattice is complete, but infinite lattices may not be complete, and hence the significance of the notion of a complete partial order [1]. Given a complete lattice $(\chi, \leq)$, we will be concerned with a special kind of a sublattice called an *interval sublattice* defined as follows. Any interval sublattice of $(\chi, \leq)$ is given by $[L, U] = \{w \in \chi : L \leq w \leq U\}$ for $L, U \in \chi$. That is, this special sublattice can be represented by two elements only. For example, the interval sublattices of $(\mathbb{R}, \leq)$ are just the familiar closed intervals on the real line. A particular instance of a partial order is the $\wedge$-*semilattice*, which is a partially ordered set in which all meet ($\wedge$) exist but all joins do not necessarily exist.

Let $(\chi, \leq)$ be a lattice with least element $\perp$ (the bottom). Then, $a \in \chi$ is called an *atom* of $(\chi, \leq)$ if $a > \perp$ and there is no element b such that $\perp < b < a$. The set of atoms of $(\chi, \leq)$ is denoted $\mathcal{A}(\chi, \leq)$.

The *power lattice* of a set $\mathcal{U}$, denoted $(\mathcal{P}(\mathcal{U}), \subseteq)$, is given by the power set of $\mathcal{U}$, $\mathcal{P}(\mathcal{U})$ (the set of all subsets of $\mathcal{U}$), ordered according to the set inclusion $\subseteq$. The meet and join of the power lattice is given by intersection and union. The bottom element is the empty set, that is, $\perp = \emptyset$, and the top element is $\mathcal{U}$ itself, that is, $\top = \mathcal{U}$. Note that $\mathcal{A}(\mathcal{P}(\mathcal{U}), \subseteq) = \mathcal{U}$. An example is illustrated in Figure 2.2. Given a set P, we denote by $|P|$ its cardinality.

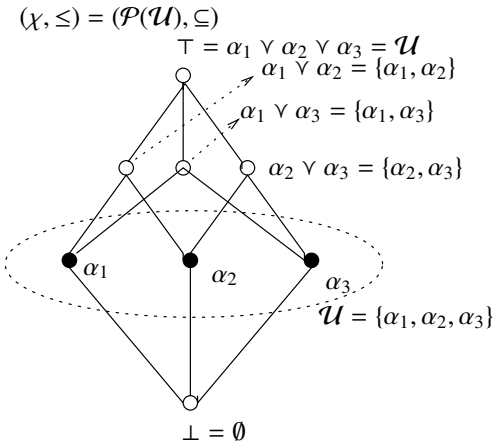


Figure 2.2: Power lattice $(\chi, \leq)$ of a set $\mathcal{U}$ composed of three elements.

Definition 2.1.1. Let $(P, \leq)$ and $(Q, \leq)$ be partially ordered sets. A map $f : P \to Q$ is

(i) an *order preserving map* if $x \leq w \implies f(x) \leq f(w)$;

(ii) an *order embedding* if $x \leq w \iff f(x) \leq f(w)$;

(iii) an *order isomorphism* if it is order embedding and it maps P *onto* Q.

Definition 2.1.2. If $(P, \leq)$ and $(Q, \leq)$ are lattices, then a map $f : P \to Q$ is said to be a *homomorphism* if f is *join-preserving* and *meet-preserving*, that is, for all $x, w \in P$ we have that $f(x \vee w) = f(x) \vee f(w)$ and $f(x \wedge w) = f(x) \wedge f(w)$.

Proposition 2.1.1. *(See [21]) If $f : P \to Q$ is a bijective homomorphism, then it is an order isomorphism.*

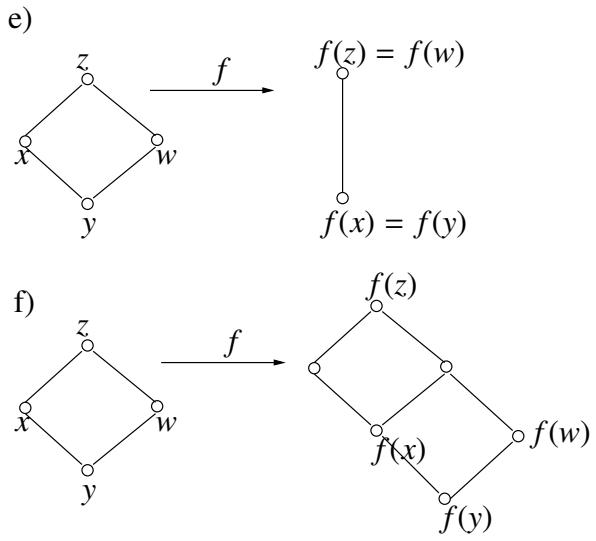


Figure 2.3: In diagram e), we show a map that is, order preserving but not order embedding. In diagram f), we show an order embedding that is, not an order isomorphism: any two elements maintain the same order relation, but in between z and w there is nothing, while in between $f(z)$ and $f(w)$ some other elements appear (it is not onto).

Every order isomorphic map faithfully mirrors the structure of P onto Q. In Figure 2.3 we show some examples.

The notion of an order preserving map can be generalized to the case in which the map is nondeterministic, that is, it maps an element to a set of possible elements. With a slight abuse of the term "order preserving" we also make the following non-standard definition.

Definition 2.1.3. Let $x, w \in \chi$, with $(\chi, \leq)$ a lattice, $x \leq w$, and $f : \chi \to \mathcal{P}(\chi)$. We say that f is *order preserving* if $\bigvee f(x) \leq \bigvee f(w)$ and $\bigwedge f(x) \leq \bigwedge f(w)$.

A partial order induces a notion of distance between elements in the space. Define the distance function on a partial order in the following way.

Definition 2.1.4. (Distance on a partial order) Let $(P, \leq)$ be a partial order. A distance d on $(P, \leq)$ is a function $d : P \times P \to \mathbb{R}$ such that the following properties are verified:

(i) $d(x, y) \geq 0$ for any $x,\ y \in P$ and $d(x, y) = 0$ if and only if $x = y$;

(ii) $d(x, y) = d(y, x)$;

(iii) if $x \leq y \leq z$ then $d(x, y) \leq d(x, z)$.

Since Chapter 6 deals with a partial order on the space of the discrete variables and with a partial order on the space of the continuous variables, it is useful to introduce the Cartesian product of two partial orders as it can be found in [1].

Definition 2.1.5. (Cartesian product of partial orders) Let $(P_1, \leq)$ and $(P_2, \leq)$ be two partial orders. Their Cartesian product is given by $(P_1 \times P_2, \leq)$, where $P_1 \times P_2 = \{(x, y) \mid x \in P_1 \text{ and } y \in P_2\}$, and $(x, y) \leq (x', y')$ if and only if $x \leq x'$ and $y \leq y'$. For any $(p_1, p_2) \in P_1 \times P_2$ the standard projections $\pi_1 : P_1 \times P_2 \to P_1$ and $\pi_2 : P_1 \times P_2 \to P_2$ are such that $\pi_1(p_1, p_2) = p_1$ and $\pi_2(p_1, p_2) = p_2$.

One can easily verify that the projection operators preserve the orders.

In this work we will also deal with *approximations* of sets and elements of a partial order. We thus give the following definition.

Definition 2.1.6. (Upper and lower approximation) Let P_1 and P_2 be two sets with $P_1 \subseteq P_2$ and $(P_2, \leq)$ a partial order. For any $x \in P_2$, we define the *lower and upper approximations* of x in P_1 as

$$a_L(x) := \max_{(P_2, \leq)} \{w \in P_1 \mid w \leq x\}$$
$$a_U(x) := \min_{(P_2, \leq)} \{w \in P_1 \mid w \geq x\}.$$

If such lower and upper approximations exist for any $x \in P_2$, then the partial order $(P_2, \leq)$ is said to be *closed with respect to* P_1.

One can verify that the lower and upper approximation functions are order preserving. This means that for any $x_1, x_2 \in P_2$ with $x_1 \leq x_2$, then $a_L(x_1) \leq a_L(x_2)$ and $a_U(x_1) \leq a_U(x_2)$. Example of lower and upper approximations are depicted in Figure 2.4.

In this section, we have given some basic definitions on partial order and lattice theory. In the next section, we introduce the class of models that we are going to consider in this work. These are transition systems with output.

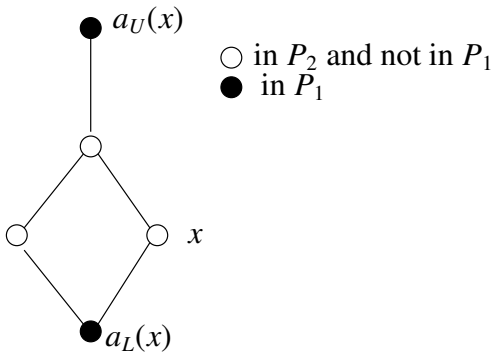


Figure 2.4: Let $P_1 \subseteq P_2$, the open circles represent elements in P_2 that are not in P_1 while the filled circles represent elements that are also in P_1.

2.2 Deterministic Transition Systems

The class of systems we are concerned with are deterministic, infinite state systems with output. The following definition introduces such a class.

Definition 2.2.1. (Deterministic transition systems) A *deterministic transition system* (DTS) is the tuple $\Sigma = (S, \mathcal{Y}, F, g)$, where

(i) S is a set of states with $s \in S$;

(ii) $\mathcal{Y}$ is a set of outputs with $y \in \mathcal{Y}$;

(iii) $F : S \to S$ is the state transition function;

(iv) $g : S \to \mathcal{Y}$ is the output function.

An execution of Σ is any sequence $\sigma = \{s(k)\}_{k\in\mathbb{N}}$ such that $s(0) \in S$ and $s(k+1) = F(s(k))$ for all $k \in \mathbb{N}$. The set of all executions of Σ is denoted $\mathcal{E}(\Sigma)$. An output sequence of Σ is denoted $y = \{y(k)\}_{k\in\mathbb{N}}$, with $y(k) = g(\sigma(k))$, for $\sigma \in \mathcal{E}(\Sigma)$.

Definition 2.2.2. Let $\Sigma = (S, \mathcal{Y}, F, g)$ be a deterministic transition system. The set $\Omega \subset S$ is the *ω^+-limit set* of Σ, denoted $\omega(\Sigma)$, if it is the smallest subset of S such that for all $\sigma = \{s(k)\}_{k\in\mathbb{N}}$

(i) if $s(k) \in \Omega$ and $s(k+1) = F(s(k))$, then $s(k+1) \in \Omega$;

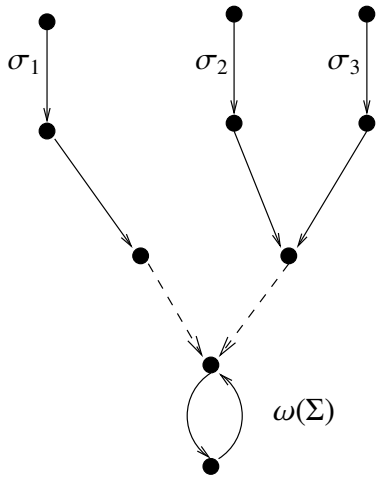


Figure 2.5: Executions σ_2 and σ_3 are weakly equivalent according to Definition 2.2.5 while σ_1 is not weakly equivalent to either σ_2 or σ_3.

(ii) for each $\sigma \in \mathcal{E}(\Sigma)$, there exists k_σ such that $\sigma(k_\sigma) \in \Omega$.

Definition 2.2.3. Given a deterministic transition system $\Sigma = (S, \mathcal{Y}, F, g)$, two executions $\sigma_1, \sigma_2 \in \mathcal{E}(\Sigma)$ are *distinguishable* if there exists a k such that $g(\sigma_1(k)) \neq g(\sigma_2(k))$.

Definition 2.2.4. (Observability) The deterministic transition system $\Sigma = (S, \mathcal{Y}, F, g)$ is said to be *observable* if any two different executions $\sigma_1, \sigma_2 \in \mathcal{E}(\Sigma)$ are distinguishable.

From this definition, we deduce that if a system Σ is observable, any two different initial states will give rise to two executions σ_1 and σ_2 with different output sequences. Thus, the initial states can be distinguished by looking at the output sequence. However, there are systems for which two different initial states cannot be distinguished, but the states at some later step can. We introduce a weaker notion of observability analogous to *detectability* [41] that accounts for this distinction.

Definition 2.2.5. Given a deterministic transition system $\Sigma = (S, \mathcal{Y}, F, g)$, two executions $\sigma_1, \sigma_2 \in \mathcal{E}(\Sigma)$ are *weakly equivalent*, denoted $\sigma_1 \sim \sigma_2$, if there exists k^* such that $\sigma_1(k^*) \notin \omega(\Sigma)$ and $\sigma_1(k) = \sigma_2(k)$ for all $k \geq k^*$.

In Figure 2.5, we show examples of equivalent and not equivalent system executions.

Definition 2.2.6. (Weak observability) A deterministic transition system $\Sigma = (S, \mathcal{Y}, F, g)$ is *weakly observable* if whenever $\sigma_1 \nsim \sigma_2$ then there is k such that $g(\sigma_1(k)) \neq g(\sigma_2(k))$.

For any system Σ, the state estimation problem is defined as follows.

Problem 2.2.1. (State estimation problem) Given Σ and any output sequence $y = \{y(k)\}_{k\in\mathbb{N}}$, determine $\{s(k)\}_{k\geq k_0}$ for some $k_0 > 0$.

In the next section, a solution to this problem, first introduced by Caines ([13, 14]), is presented.

2.3 Enumeration Approach to the Discrete State Estimation Problem

Let $\Sigma = (S, \mathcal{Y}, F, g)$, with S a finite set and $\mathcal{Y} = \{\mathcal{Y}_1, ..., \mathcal{Y}_m\}$ with $m \leq |S|$. We use a variable $\hat{s}$ to represent an estimate of s with $\hat{s} \in \mathcal{P}(S)$. Since S is composed by a finite number of elements, the estimation problem is called the discrete state estimation problem. The intention is that $\hat{s}(k)$ denotes the set of all possible values of $s(k)$ compatible with the output sequence until step k and with the system dynamics. For $k \geq 0$, $\hat{s}(k)$ is updated according to

$$\hat{s}(k+1) = F(\hat{s}(k)) \cap O_y(k+1), \qquad \hat{s}(0) = S, \tag{2.1}$$

where for any $\hat{s} \in \mathcal{P}(S)$, we define

$$F(\hat{s}) = \{s' \in S \ : \ \exists\ s \in \hat{s} \text{ with } F(s) = s'\},$$

and O_y is the *output set* and it is defined by $O_y := g^{-1}(y)$, with $g^{-1} : \mathcal{Y} \to \mathcal{P}(S)$ is the inverse of g defined as

$$g^{-1}(y) = \{s \in S \ : \ g(s) = y\}.$$

Equation (2.1) gives at each step k a set that contains all and only the states compatible with the system dynamics and with the output sequence up to step $k+1$. A picture representing this update law is in Figure 2.6.

For such an update law, the following result holds.

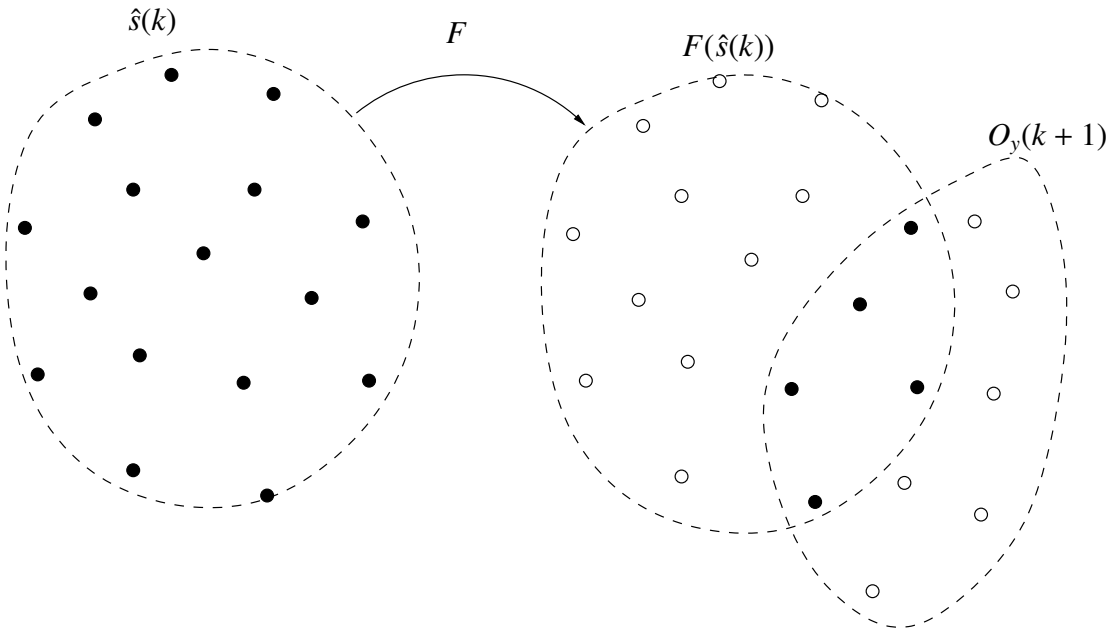


Figure 2.6: Enumeration approach to state estimation: the set of possible consistent states $\hat{s}(k)$ is mapped forward through the system dynamics F, and then $F(\hat{s}(k))$ is intersected with the set of all states compatible with the new output ($O_y(k+1)$). This procedure gives $\hat{s}(k+1)$, which is represented by the filled circles in the right diagram.

Theorem 2.3.1. *Given the system $\Sigma = (S, \mathcal{Y}, F, g)$, the update law in equation (2.1) is such that*

(i) $s(k) \in \hat{s}(k)$ *for any* $k \geq 0$ *(correctness);*

(ii) $|\hat{s}(k+1)| \leq |\hat{s}(k)|$ *(non-increasing error);*

(iii) if Σ *is (weakly) observable, then there is* $k_0 > 0$ *such that* $\hat{s}(k) = s(k)$ *for any* $k \geq k_0$ *(convergence).*

Proof. Proof of (i). This can be proved by induction argument on the step k. Briefly, $s(0) \in \hat{s}(0)$. Assume $s(k) \in \hat{s}(k)$, we prove that $s(k+1) \in \hat{s}(k+1)$. This follows from two facts. Fact 1): $s(k+1) \in g^{-1}(y(k+1))$ because $g(s(k+1)) = y(k+1)$. Fact 2): Since $s(k) \in \hat{s}(k)$, also $s(k+1) = F(s(k)) \in F(\hat{s}(k))$.

Proof of (ii). This follows directly from the following two facts. Fact 1): $|F(\hat{s}(k))| = |\hat{s}(k)|$. Fact 2): For any two sets A and B, $|A \cap B| \leq |A|$.

Proof of (iii). This can be proved by contradiction. Assume that there is no k_0 such that $\hat{s}(k) = s(k)$, then one can construct two executions of Σ, $\sigma_1 \neq \sigma_2$ such that $g(\sigma_1(k)) = g(\sigma_2(k))$ for any k. This contradicts (weak) observability. □

This proof is just a sketch. For a complete proof, the reader is deferred to [13]. This enumeration approach to state estimation will also be referred to as current location observation tree method, due to the tree implementation provided in [13].

From expression (2.1), it is clear that this approach is impracticable if the size of S is large. This is often the case in multi-agent and distributed systems, in which each agent has a set of possible states and the overall state of the system explodes combinatorially in the number of states of each agent. For example, if we have a multi-agent system composed by N agents each of which can be in n different states, the size of S is of the order of n^N, that is, $|S| \approx O(n^N)$. This means that $|S|$ grows exponentially in the number of agents, and thus the computational complexity of the enumeration approach to estimation grows exponentially as well in the number of agents for a multi-agent system. In the next chapter, we propose a methodology to overcome this state explosion problem and show an example of a multi-robot system in which the computational complexity of the estimation algorithm is $O(N)$.

Chapter 3

Construction of Discrete State Estimators on a Lattice

In the previous chapter, we have shown an enumeration approach to the discrete state estimation problem (first introduced by Caines [13]). Such an approach is however impracticable when the dimension of the state space is large as is often the case in multi-agent or distributed systems. In this chapter, we propose an alternative to the enumeration of the compatible states. In particular, a set is represented by a lower and an upper bound once it has been immersed in a lattice structure. We then keep track of the set by updating its lower and upper bounds as opposed to the list of elements it contains. As a motivating example, we introduce in Section 3.1 a multi-robot system. In Section 3.2, the state estimation problem is formulated on a lattice, and a solution is proposed in Section 3.3. Finally, the example presented in Section 3.1 is revisited, and the estimator constructed in Section 3.4. The results of this chapter appeared in [23].

3.1 Motivating Example

As a motivating example, we consider a task that represents a defensive maneuver for a robotic "capture the flag" game [20]. We do not propose to devise a strategy that addresses the full complexity of the game. Instead, we examine the following very simple *drill* or exercise that we call "RoboFlag Drill." Some number of blue robots with positions $(z_i, 0) \in \mathbb{R}^2$ (denoted by open circles) must defend their zone $\{(x, y) \in \mathbb{R}^2 \mid y \leq 0\}$ from an equal number of incoming red robots (denoted by filled circles). The positions of the red robots

are $(x_i, y_i) \in \mathbb{R}^2$. An example for 5 robots is illustrated in Figure 3.1.

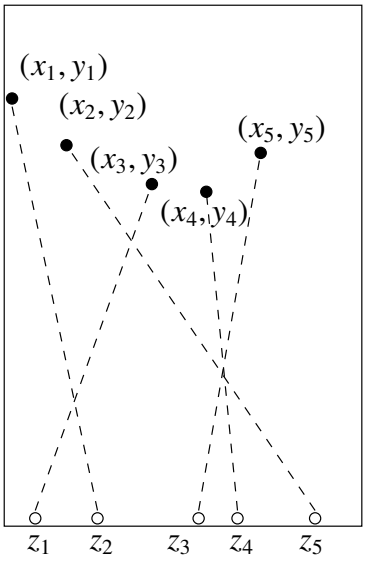


Figure 3.1: An example state of the RoboFlag Drill for 5 robots. The dashed lines represent the assignment of each blue robot to red robot. Here, the assignment is $\alpha = \{3, 1, 5, 4, 2\}$. The variables z_i denotes the position along the horizontal axis of blue robot i, and (x_i, y_i) denotes the position in the plane of red robot i.

The red robots move straight toward the blue robots' defensive zone. The blue robots are each assigned to a red robot, and they coordinate to intercept the red robots. Let N represent the number of robots in each team. The robots start with an arbitrary (bijective) assignment $\alpha : \{1, ..., N\} \rightarrow \{1, ..., N\}$, where α_i is the red robot that blue robot i is required to intercept. At each step, each blue robot communicates with its neighbors and decides to either switch assignments with its left or right neighbor or keep its assignment. It is possible to show that the α assignment reaches the equilibrium value $(1, ..., N)$ (see [35] or [34] for details). We consider the problem of estimating the current assignment α given the motions of the blue robots, which might be of interest to, for example, the red robots in that they may use such information to determine a better strategy of attack. We do not consider the problem of how they would change their strategy in this work.

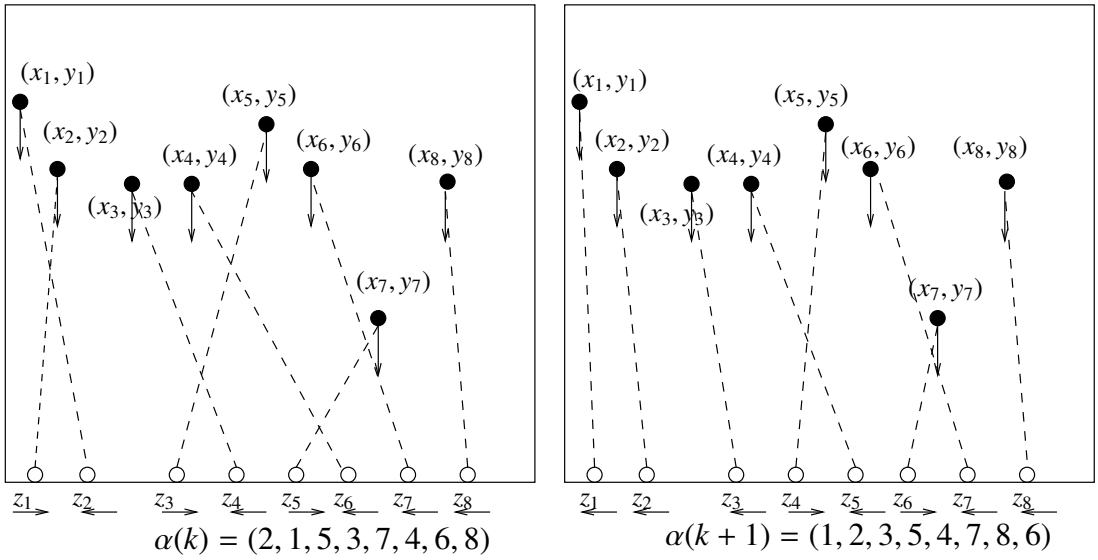


Figure 3.2: Example of the RoboFlag Drill with 8 robots per team. The dashed lines represent the assignment of each blue robot to red robot. The arrows denote the direction of motion of each robot.

The RoboFlag Drill system can be specified by the following rules:

$$y_i(k+1) = y_i(k) - \delta \quad \text{if} \quad y_i(k) \geq \delta \tag{3.1}$$

$$z_i(k+1) = z_i(k) + \delta \quad \text{if} \quad z_i(k) < x_{\alpha_i(k)} \tag{3.2}$$

$$z_i(k+1) = z_i(k) - \delta \quad \text{if} \quad z_i(k) > x_{\alpha_i(k)} \tag{3.3}$$

$$(\alpha_i(k+1), \alpha_{i+1}(k+1)) = (\alpha_{i+1}(k), \alpha_i(k)) \quad \text{if} \quad x_{\alpha_i(k)} \geq z_{i+1}(k) \wedge x_{\alpha_{i+1}(k)} \leq z_{i+1}(k), \tag{3.4}$$

where we assume $z_i \leq z_{i+1}$ and $x_i < z_i < x_{i+1}$ for all k. Also, if none of the "if" statements above are verified for a given variable, the new value of the variable is equal to the old one. This system is a slight simplification of the original system described in [34]. In such a work in fact, two close robots might decide to swap their assignments even if they are moving in the same direction, while in the present case, two close robots swap their assignments only if they are moving one toward the other. Also, in [34] the decision are taken sequentially first from the robots on the left and then from the robots on the right, and the decision are coordinated by a token that moves from left to right. In the presebt case, the decision protocol is completely decentralized.

Equation (3.4) establishes that two robots trade their assignments if the current assignments cause them to go toward each other. The question we are interested in is the following: given the evolution of the measurable quantities z, x, y, can we build an estimator that tracks on-line the value of the assignment $\alpha(k)$? The value of $\alpha \in \text{perm}(N)$ determines the discrete state, i.e., $S = \text{perm}(N)$. The discrete state α determines also what has been called in previous work the location of the system (see [4]). The number of possible locations is $N!$, that is, $|S| = N!$. This for $N \geq 8$ renders prohibitive the application of location observers based on the current location observation tree as described in [13] (revised in Chapter 2) and used in [4], [22]. At each step, the set of possible α values compatible with the current output and with the previously seen outputs can be so large as to render impractical its computation. As an example, we consider the situation depicted in Figure 3.2 (left)

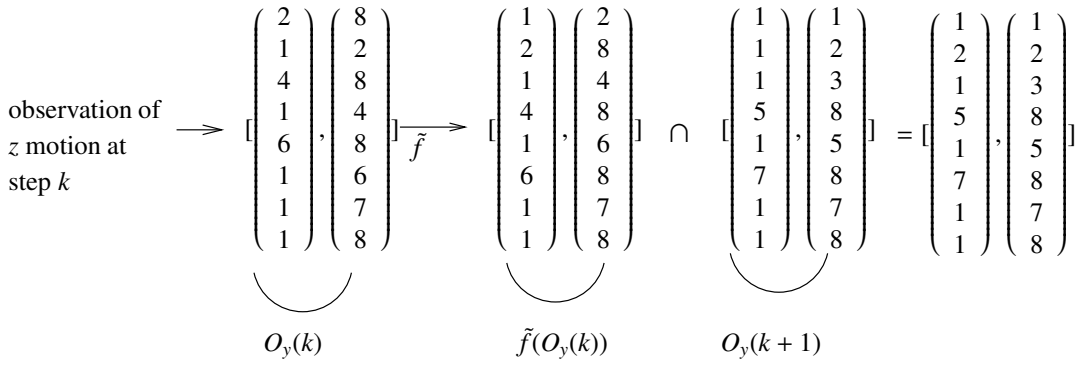


Figure 3.3: The observation of the z motion at step k gives the set of possible α, $O_y(k)$. At each step, the set is described by the lower and upper bounds of an *interval sublattice* in an appropriately defined lattice. Such set is then mapped through the system dynamics ($\tilde{f}$) to obtain at step $k+1$ the set of α that are compatible also with the observation at step k. Such a set is then intersected with $O_y(k+1)$, which is the set of α compatible with the z motion observed at step $k+1$.

where $N = 8$. We see the blue robots $1, 3, 5$ going right and the others going left. From equations (3.2)–(3.3) with $x_i < z_i < x_{i+1}$ we deduce that the set of all possible $\alpha \in \text{perm}(N)$ compatible with this observation is such that $\alpha_i \geq i+1$ for $i \in \{1, 2, 3\}$ and $\alpha_i \leq i$ for $i \in \{2, 4, 6, 7, 8\}$. The size of this set is 40320. According to the enumeration methods presented in Section 2.3, this set needs to be mapped forward through the dynamics of the system to see what are the values of α at the next step that correspond to this output.

Such a set is then intersected with the set of α values compatible with the new observation. To overcome the complexity issue that comes from the need of listing 40320 elements for performing such operations, we propose to represent a set by a lower L and an upper U elements according to some partial order. Then, we can perform the previously described operations only on L and U, two elements instead of 40320. This idea is developed in the following paragraph.

For this example, we can view $\alpha \in \mathbb{N}^N$. The set of possible assignments compatible with the observation of the z motion deduced from the equations (3.2)–(3.3), denoted $O_y(k)$, can be represented as an interval with the order established component-wise, see the diagram in Figure 3.3. The function $\tilde{f}$ that maps such a set forward, specified by the equations (3.4) with the assumption that $x_i < z_i < x_{i+1}$, simply swaps two adjacent robot assignments if these cause the two robots to move toward each other. Thus, it maps the set $O_y(k)$ to the set $\tilde{f}(O_y(k))$ shown in Figure 3.3, which can still be represented as an interval. When the new output measurement becomes available (Figure 3.2, right) we obtain the new set $O_y(k+1)$ reported in Figure 3.3. The sets $\tilde{f}(O_y(k))$ and $O_y(k+1)$ can be intersected by simply computing the supremum of their lower bounds and the infimum of their upper bounds. This idea is also explained in Figure 3.4. This way, we obtain the system that updates L and U, being L and U the lower and upper bounds of the set of all possible α compatible with the output sequence:

$$\begin{aligned} L(k+1) &= \tilde{f}(\sup(L(k), \inf O_y(k))) \\ U(k+1) &= \tilde{f}(\inf(U(k), \sup O_y(k))). \end{aligned} \tag{3.5}$$

The variables $L(k)$ and $U(k)$ represent the lower and upper bound, respectively, of the set $\hat{s}$ computed in equation (2.1). The computational burden of this implementation is of the order of N if N is the number of robots. This computational burden is to be compared to $N!$, which is the computation requirement that we have with the enumeration approach as noticed earlier in this section.

As it will be shown in detail in this thesis, the update laws in equations (3.5) have, among others, the property that $[L(k), U(k)] \cap \text{perm}(N)$ tends to $\alpha(k)$. Letting $V(k) =$

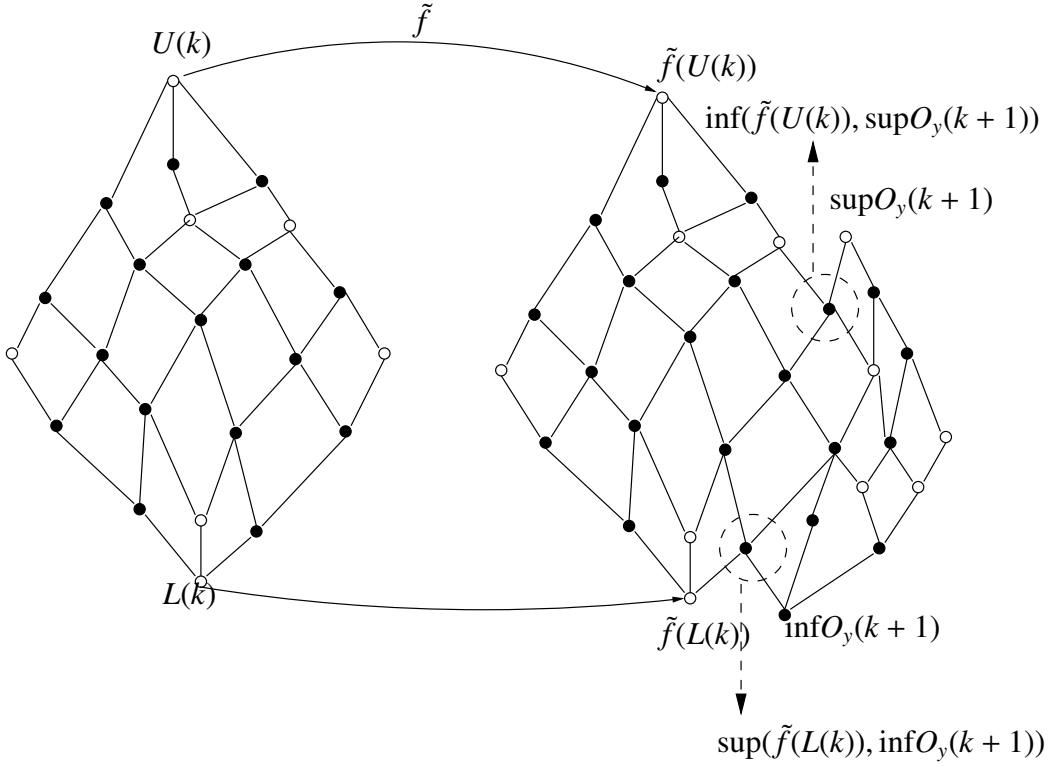


Figure 3.4: Lattice approach to state estimation. The set of possible consistent states $\hat{s}(k)$ is represented by a lower and an upper bound $L(k)$ and $U(k)$, once the set has been immersed into a lattice. Then, the function $\tilde{f}$ is computed on $L(k)$ and $U(k)$, only. The intersection with the output set at step k, $O_y(k+1) = [\inf O_y(k+1), \sup O_y(k+1)]$, is computed by computing the supremum and infimum of the set intersection. Its supremum is $\inf(\tilde{f}(U(k)), \sup O_y(k+1))$ and its infimum is $\sup(\tilde{f}(L(k)), \inf O_y(k+1))$.

$|[L(k), U(k)] \cap \text{perm}(N)|$, Figure 3.5 shows convergence plots $V(k)$ for the estimator compared to the convergence plots $E(k) = 1/N \sum_{i=1}^{N} |\alpha_i(k) - i|$ of the assignment protocol to its equilibrium $(1, ..., N)$.

This example gives an idea of how complexity issues can be overcome with the aid of some partial order structure. In particular, the function $\tilde{f}$ has the property of preserving the interval structure of the sets of interest: this is a key property that allows to use of upper lower bounds only for computation purposes. In a more general setting, one would like to know what are the system properties that allow such simplifications. By using partial order theory, we address this question.

In the following section, we restrict the class of systems introduced in the previous chapter to those in which the continuous variables are measurable. The discrete state esti-

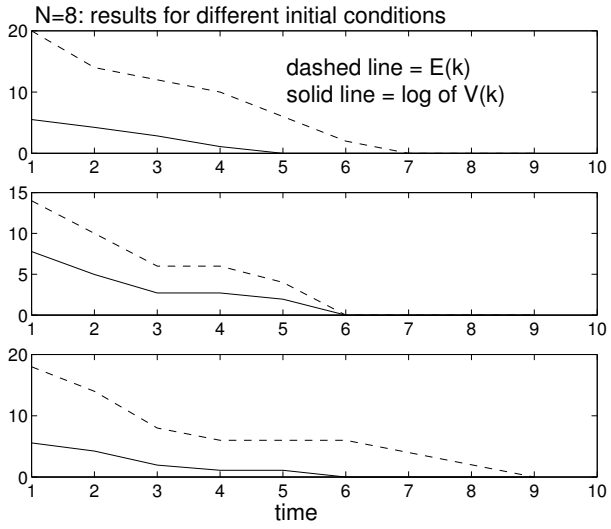


Figure 3.5: Convergence plots for the estimator (V(k)) compared to the convergence plot of the assignment protocol to its equilibrium (E(k)).

mation problem is then stated as the problem of finding suitable update laws for the upper and lower bounds of the set of all possible discrete variable values compatible with the output sequence. A solution to this problem is proposed in Theorem 3.3.1.

3.2 Problem Formulation

The deterministic transition systems Σ we defined in the previous chapter are quite general. In this section, we restrict our attention to systems with a specific structure. In particular, for a system $\Sigma = (S, \mathcal{Y}, F, g)$ we suppose that

(i) $S = \mathcal{U} \times \mathcal{Z}$ with $\mathcal{U}$ a finite set and $\mathcal{Z}$ a finite dimensional space;

(ii) $F = (f, h)$, where $f : \mathcal{U} \times \mathcal{Z} \to \mathcal{U}$ and $h : \mathcal{U} \times \mathcal{Z} \to \mathcal{Z}$;

(iii) $y = g(\alpha, z) := z$, where $\alpha \in \mathcal{U}$, $z \in \mathcal{Z}$, $y \in \mathcal{Y}$, and $\mathcal{Y} = \mathcal{Z}$.

The set $\mathcal{U}$ is a set of logic states and $\mathcal{Z}$ is a set of measured states or physical states, as one might find in a robot system. In the case of the example given in Section 3.1, $\mathcal{U} = \text{perm}(N)$ and $\mathcal{Z} = \mathbb{R}^N$, the function f is represented by equations (3.4) and the function h is represented by equations (3.2)–(3.3). In the sequel, we will denote this class

of deterministic transition systems by $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ where we associate to the tuple $(\mathcal{U}, \mathcal{Z}, f, h)$, the equations:

$$\begin{aligned} \alpha(k+1) &= f(\alpha(k), z(k)) \\ z(k+1) &= h(\alpha(k), z(k)) \\ y(k) &= z(k), \end{aligned} \tag{3.6}$$

where $\alpha \in \mathcal{U}$ and $z \in \mathcal{Z}$. An execution of the system Σ in equations (3.6) is a sequence $\sigma = \{\alpha(k), z(k)\}_{k\in\mathbb{N}}$. The output sequence is $\{y(k)\}_{k\in\mathbb{N}} = \{z(k)\}_{k\in\mathbb{N}}$. Given an execution σ of the system Σ, we denote the α and z sequences corresponding to such an execution by $\{\sigma(k)(\alpha)\}_{k\in\mathbb{N}}$ and $\{\sigma(k)(z)\}_{k\in\mathbb{N}}$, respectively.

From the measurement of the output sequence, which in our case coincides with the evolution of the continuous variables, we want to construct a discrete state estimator: a system $\hat{\Sigma}$ that takes as input the values of the measurable variables and asymptotically tracks the value of the variable α. We thus define in the following definition a deterministic transition system with input.

Definition 3.2.1. (Deterministic transition system with input) A deterministic transition system with input is a tuple $(S, \mathcal{I}, \mathcal{Y}, F, g)$ in which

(i) S is a set of states;

(ii) $\mathcal{I}$ is a set of inputs;

(iii) $\mathcal{Y}$ is a set of outputs;

(iv) $F : S \times \mathcal{I} \to S$ is a transition function;

(v) $g : S \to \mathcal{Y}$ is an output function.

In Problem 3.2.1 below, we specify what the elements of this tuple are when the DTS with input is a discrete state estimator of a DTS $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$. First, note that the set $\mathcal{U}$ does not have a natural metric associated with it. As a consequence, a way to track the value of α is to list, at each step k, the set of all possible α values that are compatible with

the observation and with the system dynamics given in (3.6). This enumeration approach has been shown in Section 2.3, in which the estimate is a list of possible values that the estimator has to update when a new measurement becomes available. This method leads to computational issues when the set to be listed is large.

In this chapter, an alternative to simply maintaining a list of all possible values for α is proposed. We find a representation of the set so that the estimator updates the representation of the set rather than the whole set itself. In particular, if the set $\mathcal{U}$ can be immersed in a larger set χ whose elements can be related by an order relation $\leq$, we could represent a subset of $(\chi, \leq)$ as an interval sublattice $[L, U]$. Let "id" denote the identity operator. We formulate the discrete state estimation problem on a lattice as follows.

Problem 3.2.1. (Discrete state estimator on a lattice) Given the deterministic transition system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$, find a deterministic transition system with input $\hat{\Sigma} = (\chi \times \chi, \mathcal{Z} \times \mathcal{Z}, \chi \times \chi, (f_1, f_2), \text{id})$, with $f_1 : \chi \times \mathcal{Z} \times \mathcal{Z} \to \chi$, $f_2 : \chi \times \mathcal{Z} \times \mathcal{Z} \to \chi$, $\mathcal{U} \subseteq \chi$, with $(\chi, \leq)$ a lattice, represented by the equations

$$\begin{aligned} L(k+1) &= f_1(L(k), y(k), y(k+1)) \\ U(k+1) &= f_2(U(k), y(k), y(k+1)), \end{aligned}$$

with $L(k) \in \chi$, $U(k) \in \chi$, $L(0) := \bigwedge \chi$, $U(0) := \bigvee \chi$, such that

(i) $L(k) \leq \alpha(k) \leq U(k)$ (correctness);

(ii) $|[L(k+1), U(k+1)]| \leq |[L(k), U(k)]|$ (non-increasing error);

(iii) There exists $k_0 > 0$ such that for any $k \geq k_0$ we have $[L(k), U(k)] \cap \mathcal{U} = \alpha(k)$ (convergence).

3.3 Problem Solution

For finding a solution to Problem 3.2.1, we need to find the functions f_1 and f_2 defined on a lattice $(\chi, \leq)$ such that $\mathcal{U} \subseteq \chi$ for some finite lattice χ. We propose in the following

definitions a way of extending a system Σ defined on $\mathcal{U}$ to a system $\tilde{\Sigma}$ defined on χ with $\mathcal{U} \subseteq \chi$. Moreover, as we have seen in the motivating example, we want to represent the set of possible α values compatible with an output measurement as an interval sublattice in $(\chi, \leq)$. We thus define the $\tilde{\Sigma}$ transition classes, with each transition class corresponding to a set of values in χ compatible with an output measurement. We define the partial order $(\chi, \leq)$ and the system $\tilde{\Sigma}$ to be interval compatible if such equivalence classes are interval sublattices and $\tilde{\Sigma}$ preserves their structure. On the basis of such notions, Theorem 3.3.1 below gives a possible solution to Problem 3.2.1.

Definition 3.3.1. (Extended system) Given the deterministic transition system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$, an *extension of* Σ *on* χ, with $\mathcal{U} \subseteq \chi$ and $(\chi, \leq)$ a finite lattice, is any system $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$, such that

(i) $\tilde{f} : \chi \times \mathcal{Z} \to \chi$ and $\tilde{f}|_{\mathcal{U}\times\mathcal{Z}} = f$;

(ii) $\tilde{h} : \chi \times \mathcal{Z} \to \mathcal{Z}$ and $\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$.

Definition 3.3.2. (Transition sets) Let $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ be a deterministic transition system. The non empty sets $T_{(z^1,z^2)}(\tilde{\Sigma}) = \{w \in \chi \mid z^2 = \tilde{h}(w, z^1)\}$, for $z^1, z^2 \in \mathcal{Z}$, are named the $\tilde{\Sigma}$*-transition sets*.

Each $\tilde{\Sigma}$-transition set contains all of $w \in \chi$ values that allow the transition from z^1 to z^2 through $\tilde{h}$. It will be also useful to define the transition class $\mathcal{T}_i(\tilde{\Sigma})$, which corresponds to multiple transition sets, as transition sets obtained by different pairs (z^1, z^2) can define the same set in χ.

Definition 3.3.3. (Transition classes) The set $\mathcal{T}(\tilde{\Sigma}) = \{\mathcal{T}_1(\tilde{\Sigma}), ..., \mathcal{T}_M(\tilde{\Sigma})\}$, with $\mathcal{T}_i(\tilde{\Sigma})$ such that

(i) For any $\mathcal{T}_i(\tilde{\Sigma}) \in \mathcal{T}(\tilde{\Sigma})$ there are $z^1, z^2 \in \mathcal{Z}$ such that $\mathcal{T}_i(\tilde{\Sigma}) = T_{(z^1,z^2)}(\tilde{\Sigma})$;

(ii) For any $T_{(z^1,z^2)}(\tilde{\Sigma})$ there is $j \in \{1, ..., M\}$ such that $T_{(z^1,z^2)}(\tilde{\Sigma}) = \mathcal{T}_j(\tilde{\Sigma})$;

is the *set of* $\tilde{\Sigma}$*-transition classes*.

Note that $T_{(z^1,z^2)}$ and $T_{(z^3,z^4)}$ might be the same set even if $(z^1, z^2) \neq (z^3, z^4)$: in the RoboFlag Drill example introduced in Section 3.1, if robot j is moving right, the set of possible values of α_j is $[j+1, N]$ independently of the values of $z_j(k)$. Thus, $T_{(z^1,z^2)}$ and $T_{(z^3,z^4)}$ can define the same set that we call $\mathcal{T}_i(\tilde{\Sigma})$ for some i. Also, the transition classes $\mathcal{T}_i(\tilde{\Sigma})$ are not necessarily equivalence classes as they might not be pairwise disjoint. However, for the RoboFlag Drill it is the case that the transition classes are pairwise disjoint, and thus they partition the lattice $(\chi, \leq)$ in equivalence classes.

Definition 3.3.4. (Output set) Given the extension $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ of the deterministic transition system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ on the lattice $(\chi, \leq)$, and given an output sequence $\{y(k)\}_{k\in\mathbb{N}}$ of Σ, the set

$$O_y(k) := \{w \in \chi \mid \tilde{h}(w, y(k)) = y(k+1)\}$$

is the *output set* at step k.

Note that by definition, for any k, $O_y(k) = T_{(y(k),y(k+1))}(\tilde{\Sigma})$, and thus it is equal to $\mathcal{T}_i(\tilde{\Sigma})$ for some $i \in \{1, ..., M\}$. The output set at step k is the set of all possible w values that are compatible with the pair $(y(k), y(k+1))$. By definition of the extended functions ($\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$), this output set contains also all of the values of α compatible with the same output pair.

Definition 3.3.5. (Interval compatibility) Given the extension $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ of the system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ on the lattice $(\chi, \leq)$, the pair $(\tilde{\Sigma}, (\chi, \leq))$ is said to be *interval compatible* if

(i) each $\tilde{\Sigma}$-transition class, $\mathcal{T}_i(\tilde{\Sigma}) \in \mathcal{T}(\tilde{\Sigma})$, is an interval sublattice of $(\chi, \leq)$:

$$\mathcal{T}_i(\tilde{\Sigma}) = [\bigwedge \mathcal{T}_i(\tilde{\Sigma}), \bigvee \mathcal{T}_i(\tilde{\Sigma})];$$

(ii) $\tilde{f} : (\mathcal{T}_i(\tilde{\Sigma}), z) \to [\tilde{f}(\bigwedge \mathcal{T}_i(\tilde{\Sigma}), z), \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}), z)]$ is an order isomorphism for any $i \in \{1, ..., M\}$ and for any $z \in \mathcal{Z}$.

The following theorem gives the main result, which proposes a solution to Problem 3.2.1.

Theorem 3.3.1. *Assume that the deterministic transition system* $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ *is observable. If there is a lattice* $(\chi, \leq)$*, such that the pair* $(\tilde{\Sigma}, (\chi, \leq))$ *is interval compatible, then the deterministic transition system with input* $\hat{\Sigma} = (\chi \times \chi, \mathcal{Z} \times \mathcal{Z}, \chi \times \chi, (f_1, f_2), id)$ *with*

$$\begin{aligned} f_1(L(k), y(k), y(k+1)) &= \tilde{f}\left(L(k) \vee \bigwedge O_y(k), y(k)\right) \\ f_2(U(k), y(k), y(k+1)) &= \tilde{f}\left(U(k) \wedge \bigvee O_y(k), y(k)\right) \end{aligned}$$

solves Problem 3.2.1.

Proof. In order to prove the statement of the theorem, we need to prove that the system

$$\begin{aligned} L(k+1) &= \tilde{f}(L(k) \vee \bigwedge O_y(k), y(k)) \\ U(k+1) &= \tilde{f}(U(k) \wedge \bigvee O_y(k), y(k)) \end{aligned} \tag{3.7}$$

with $L(0) = \bigwedge \chi$, $U(0) = \bigvee \chi$ is such that properties (i)–(iii) of Problem 3.2.1 are satisfied. For simplicity of notation, we omit the dependence of $\tilde{f}$ on its second argument.

Proof of (i): This is proved by induction on k. Base case: for $k = 0$ we have that $L(0) = \bigwedge \chi$ and that $U(0) = \bigvee \chi$, so that $L(0) \leq \alpha(0) \leq U(0)$. Induction step: we assume that $L(k) \leq \alpha(k) \leq U(k)$ and we show that $L(k+1) \leq \alpha(k+1) \leq U(k+1)$. Note that $\alpha(k) \in O_y(k)$. This, along with the assumption of the induction step, implies that

$$L(k) \vee \bigwedge O_y(k) \leq \alpha(k) \leq U(k) \wedge \bigvee O_y(k).$$

This last relation also implies that there is x such that $x \geq L(k) \vee \bigwedge O_y(k)$ and $x \leq \bigvee O_y(k)$. This in turn implies that

$$L(k) \vee \bigwedge O_y(k) \leq \bigvee O_y(k).$$

This in turn implies that $L(k) \vee \bigwedge O_y(k) \in O_y(k)$. Because of this, because (by analogous reasonings) $U(k) \wedge \bigvee O_y(k) \in O_y(k)$, and because the pair $(\tilde{\Sigma}, (\chi, \leq))$ is interval compatible, we can use the isomorphic property of $\tilde{f}$ (property (ii) of Definition 3.3.5), which leads to

$$\tilde{f}(L(k) \vee \bigwedge O_y(k)) \leq \alpha(k+1) \leq \tilde{f}(U(k) \wedge \bigvee O_y(k)).$$

This relationship combined with equation (3.7) proves (i).

Proof of (ii): This can be shown by proving that for any $w \in [L(k+1), U(k+1)]$ there is $z \in [L(k), U(k)]$ such that $w = \tilde{f}(z)$. By equation (3.7), $w \in [L(k+1), U(k+1)]$ implies that

$$\tilde{f}(L(k) \vee \bigwedge O_y(k)) \leq w \leq \tilde{f}(U(k) \wedge \bigvee O_y(k)). \tag{3.8}$$

In addition, we have that

$$\bigwedge O_y(k) \leq L(k) \vee \bigwedge O_y(k)$$

and

$$U(k) \wedge \bigvee O_y(k) \leq \bigvee O_y(k).$$

Because the pair $(\tilde{\Sigma}, (\chi, \leq))$ is interval compatible, by virtue of the isomorphic property of $\tilde{f}$ (property (ii) of Definition 3.3.5), we have that

$$\tilde{f}(\bigwedge O_y(k)) \leq \tilde{f}(L(k) \vee \bigwedge O_y(k))$$

and

$$\tilde{f}(U(k) \wedge \bigvee O_y(k)) \leq \tilde{f}(\bigvee O_y(k)).$$

This, along with relation (3.8), implies that

$$w \in [\tilde{f}(\bigwedge O_y(k)), \tilde{f}(\bigvee O_y(k))].$$

From this, using again the order isomorphic property of $\tilde{f}$, we deduce that there is $z \in O_y(k)$ such that $w = \tilde{f}(z)$. This with relation (3.8) implies that

$$L(k) \vee \bigwedge O_y(k) \leq z \leq U(k) \wedge \bigvee O_y(k),$$

which in turn implies that $z \in [L(k), U(k)]$.

Proof of (iii): We proceed by contradiction. Thus, assume that for any k_0 there exists a $k \geq k_0$ such that $\{\alpha(k), \beta_k\} \subseteq [L(k), U(k)] \cap \mathcal{U}$ for some $\beta_k \neq \alpha(k)$ and $\beta_k \in \mathcal{U}$. By the proof of part (ii) we also have that β_k is such that $\beta_k = \tilde{f}(\beta_{k-1})$ for some $\beta_{k-1} \in [L(k-1), U(k-1)]$.

We want to show that in fact $\beta_{k-1} \in [L(k-1), U(k-1)] \cap \mathcal{U}$. If this is not the case, we can construct an infinite sequence $\{k_i\}_{i\in\mathbb{N}^+}$ such that $\beta_{k_i} \in [L(k_i), U(k_i)] \cap \mathcal{U}$ with $\beta_{k_i} = \tilde{f}(\beta_{k_i-1})$ and $\beta_{k_i-1} \in [L(k_i - 1), U(k_i - 1)] \cap (\chi - \mathcal{U})$. Notice that $|[L(k_1 - 1), U(k_1 - 1)] \cap (\chi - \mathcal{U})| = M < \infty$. Also, we have

$$|[L(k_1), U(k_1)] \cap (\chi - \mathcal{U})| < |[L(k_1 - 1), U(k_1 - 1)] \cap (\chi - \mathcal{U})|.$$

This is due to the fact that $\tilde{f}(\beta_{k_1-1}) \notin [L(k_1), U(k_1)] \cap (\chi - \mathcal{U})$, and to the fact that each element in $[L(k_1), U(k_1)] \cap (\chi - \mathcal{U})$ comes from one element in $[L(k_1-1), U(k_1-1)] \cap (\chi - \mathcal{U})$ (proof of (ii) and because $\mathcal{U}$ is invariant under $\tilde{f}$). Thus we have a strictly decreasing sequence of natural numbers $\{|[L(k_i - 1), U(k_i - 1)] \cap (\chi - \mathcal{U})|\}$ with initial value M. Since M is finite, we reach the contradiction that $|[L(k_i - 1), U(k_i - 1)] \cap (\chi - \mathcal{U})| < 0$ for some i. Therefore, $\beta_{k-1} \in [L(k - 1), U(k - 1)] \cap \mathcal{U}$.

Thus for any k_0 there is $k \geq k_0$ such that $\{\alpha(k), \beta_k\} \subseteq [L(k), U(k)] \cap \mathcal{U}$, with $\beta_k = f(\beta_{k-1})$ for some $\beta_{k-1} \in [L(k - 1), U(k - 1)] \cap \mathcal{U}$. Also, from the proof of part (ii) we have that $\beta_{k-1} \in O_y(k - 1)$. As a consequence, there exists $\bar{k} > 0$ such that $\{\beta_{k-1}, z(k - 1)\}_{k\geq\bar{k}} = \sigma_1$ and $\{\alpha(k - 1), z(k - 1)\}_{k\geq\bar{k}} = \sigma_2$ are two executions of Σ sharing the same output. This contradicts the observability assumption. □

The following corollary is a consequence of Theorem 3.3.1 in the case in which the extended system $\tilde{\Sigma}$ is observable.

Corollary 3.3.1. *If the extended system $\tilde{\Sigma}$ of an observable system Σ is observable, then the estimator $\hat{\Sigma}$ given in Theorem 3.3.1 solves Problem 3.2.1 with $L(k) = U(k) = \alpha(k)$ for $k \geq k_0$.*

Proof. The proof proceeds by contradiction. Assume that for any $k_0 \geq 0$ there is $k \geq k_0$ such that $\{\alpha(k), \beta_k\} \subseteq [L(k), U(k)]$ for some β_k. By the proof of (ii) of Theorem 3.3.1, we have that $\beta_k = \tilde{f}(\beta_{k-1})$ for $\beta_{k-1} \in [L(k - 1), U(k - 1)]$ and $\beta_{k-1} \in O_y(k - 1)$. Thus, $\sigma_1 = \{\beta_{k-1}, z(k - 1)\}_{k\in\mathbb{N}}$ and $\sigma_2 = \{\alpha(k - 1), z(k - 1)\}_{k\in\mathbb{N}}$ are two executions of $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ that share the same output sequence. This contradicts the observability of the system $\tilde{\Sigma}$. □

An example in which the Theorem 3.3.1 holds but the Corollary 3.3.1 does not is provided by the RoboFlag Drill introduced in Section 3.1. In fact, if we allow the assignments to be in $\mathbb{N}^N$ instead of being in the set of permutation of N elements, there are different executions compatible with the same output sequence.

3.4 Example: The RoboFlag Drill

The RoboFlag Drill has been described in Section 3.1. In this section, we revisit the example by showing first that it is observable with measurable variables z, and then by finding a lattice and a system extension that can be used for constructing the estimator proposed in Theorem 3.3.1.

3.4.1 System Specification

For completeness, we report here the system specification. The red robot dynamics are described by the N rules

$$y_i(k+1) = y_i(k) - \delta \quad \text{if} \quad y_i(k) \geq \delta \tag{3.9}$$

for $i \in \{1, ..., N\}$. These state simply that the red robots move a distance δ toward the defensive zone at each step. The blue robot dynamics are described by the $2N$ rules

$$\begin{aligned} z_i(k+1) &= z_i(k) + \delta \quad \text{if} \quad z_i(k) < x_{\alpha_i(k)} \\ z_i(k+1) &= z_i(k) - \delta \quad \text{if} \quad z_i(k) > x_{\alpha_i(k)} \end{aligned} \tag{3.10}$$

for $i \in \{1, ..., N\}$. For the blue robots we assume that initially $z_i \in [z_{min}, z_{max}]$ and $z_i < z_{i+1}$ and that $x_i < z_i < x_{i+1}$ for all time. The assignment protocol dynamics is defined by

$$(\alpha_i(k+1), \alpha_{i+1}(k+1)) = (\alpha_{i+1}(k), \alpha_i(k)) \quad \text{if} \quad x_{\alpha_i(k)} \geq z_{i+1}(k) \wedge x_{\alpha_{i+1}(k)} \leq z_{i+1}(k), \tag{3.11}$$

which is a modification of the protocol presented in [34], since two adjacent robots switch assignments only if they are moving one toward the other. We define $x = (x_1, ..., x_N)$, $z = (z_1, ..., z_N)$, and $\alpha = (\alpha_1, ..., \alpha_N)$. The complete RoboFlag specification is then given by the program given in rules (3.9)–(3.11). An example with 5 robots is illustrated in Figure 3.6. In particular the rules in (3.10) model the function $h : \mathcal{U} \times \mathcal{Z} \to \mathcal{Z}$ that updates

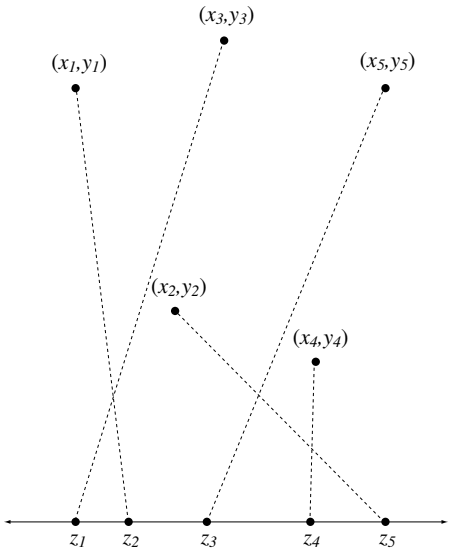


Figure 3.6: An example state of the RoboFloag Drill for 5 robots. Here, $\alpha = \{3, 1, 5, 4, 2\}$.

the continuous variables, and the rules in (3.11) model the function $f : \mathcal{U} \times \mathcal{Z} \to \mathcal{U}$ that updates the discrete variables. In this example, we have $\mathcal{U} = \text{perm}(N)$ the set of permutations of N elements, and $\mathcal{Z} = \mathbb{R}^N$. Thus, the RoboFlag system is given by $\Sigma = \mathcal{S}(\text{perm}(N), \mathbb{R}^N, f, h)$, and the variables $z \in \mathbb{R}^N$ are measured.

Problem 3.4.1. RoboFlag Drill Observation Problem. Given initial values for x and y and the values of z corresponding to an execution of $\Sigma = \mathcal{S}(\text{perm}(N), \mathbb{R}^N, f, h)$, determine the value of α during that execution.

Before constructing the estimator for the system $\Sigma = \mathcal{S}(\text{perm}(N), \mathbb{R}^N, f, h)$, we show in the following proposition that such a system is observable.

Proposition 3.4.1. *The system $\Sigma = \mathcal{S}(perm(N), \mathbb{R}^N, f, h)$ represented by the rules (3.10) and (3.11) with measurable variables z is observable.*

Proof. Given any two executions σ_1 and σ_2 of Σ, for proving observability, it is enough to show that if $\{\sigma_1(k)(\alpha)\}_{k\in\mathbb{N}} \neq \{\sigma_2(k)(\alpha)\}_{k\in\mathbb{N}}$, then $\{\sigma_1(k)(z)\}_{k\in\mathbb{N}} \neq \{\sigma_2(k)(z)\}_{k\in\mathbb{N}}$. Since the measurable variables are the z_i's, their direction of motion is also measurable. Thus, we consider the vector of directions of motion of the z_i as output. Let $g(\sigma(k))$ denote such a vector at step k for the execution σ. It is enough to show that there is a k such that $g(\sigma_1(k)) \neq g(\sigma_2(k))$. Note that, in any execution of Σ, the α trajectory reaches the equilibrium value $[1, ..., N]$, and therefore there is a step $\bar{k}$ at which $f(\sigma_1(\bar{k})) = f(\sigma_2(\bar{k}))$ and $\sigma_1(\bar{k})(\alpha) \neq \sigma_2(\bar{k})(\alpha)$. As a consequence the system is observable if $g(\sigma_1(\bar{k})) \neq g(\sigma_2(\bar{k}))$. Therefore it is enough to prove that for any $\alpha \neq \beta$, for $\alpha, \beta \in \mathcal{U}$, we have $g(\alpha, z) = g(\beta, v) \implies f(\alpha, z) \neq f(\beta, v)$, where $z, v \in \mathbb{R}^N$. Thus, $g(\alpha) = g(\beta)$ by (3.10) implies that (1) $z_i < x_{\alpha_i} \iff v_i < x_{\beta_i}$ and (2) $z_i \geq x_{\alpha_i} \iff v_i \geq x_{\beta_i}$. This implies that $x_{\alpha_i} \geq z_{i+1} \wedge x_{\alpha_{i+1}} \leq z_{i+1} \Leftrightarrow x_{\beta_i} \geq v_{i+1} \wedge x_{\beta_{i+1}} \leq v_{i+1}$. By denoting $\alpha' = f(\alpha, z)$ and $\beta' = f(\beta, v)$, we have that $(\alpha'_i, \alpha'_{i+1}) = (\alpha_{i+1}, \alpha_i) \Leftrightarrow (\beta'_i, \beta'_{i+1}) = (\beta_{i+1}, \beta_i)$. Hence if there exists an i such that $\alpha_i \neq \beta_i$, then there exists a j such that $\alpha'_j \neq \beta'_j$, and therefore $f(\alpha, z) \neq f(\beta, v)$. □

In the following subsection, the formal estimator construction is presented.

3.4.2 RoboFlag Drill Estimator

We have shown that the RoboFlag system $\Sigma = \mathcal{S}(\text{perm}(N), \mathbb{R}^N, f, h)$ represented by the rules (3.10) and (3.11) with measurable variables z is observable. In this section, we construct the estimator proposed in Theorem 3.3.1 in order to estimate and track the value of the assignment α in any execution. To accomplish this, we need to find a lattice $(\chi, \leq)$ in which to immerse the set $\mathcal{U}$ and an extension $\tilde{\Sigma}$ of the system Σ to χ, so that the pair $(\tilde{\Sigma}, (\chi, \leq))$ is interval compatible.

We first construct a lattice $(\chi, \leq)$ and the extended system $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ such that $(\tilde{\Sigma}, (\chi, \leq))$ is interval compatible. We choose as χ the set of vectors in $\mathbb{N}^N$ with coordinates $x_i \in [1, N]$, that is,

$$\chi = \{x \in \mathbb{N}^N : x_i \in [1, N]\}. \tag{3.12}$$

For the elements in χ, we use the vector notation, that is, $x = (x_1, ..., x_N)$. The partial order

that we choose on such a set is given by

$$\forall x, w \in \chi,\ x \leq w \text{ if } x_i \leq w_i\ \forall i. \tag{3.13}$$

As a consequence, the join and the meet between any two elements in χ are given by

$$\forall\ x, w\ \in \chi,\ v = x \vee w \text{ if } v_i = \max\{x_i, w_i\},$$
$$\forall\ x, w\ \in \chi,\ v = x \wedge w \text{ if } v_i = \min\{x_i, w_i\}.$$

With this choice,we have $\bigvee \chi = (N, ..., N)$ and $\bigwedge \chi = (1, ..., 1)$. The pair $(\chi, \leq)$ with the order defined by (3.13) is clearly a lattice. The set $\mathcal{U}$ is the set of all permutations of N elements and it is a subset of χ. All of the elements in $\mathcal{U}$ form an anti-chain of the lattice, that is, any two elements of $\mathcal{U}$ are not related by the order in $(\chi, \leq)$. In the reminder of this section, we will denote by w the variables in χ not specifying if it is in $\mathcal{U}$, and we will denote by α the variables in $\mathcal{U}$.

The function $h : \text{perm}(N) \times \mathbb{R}^N \to \mathbb{R}^N$ can be naturally extended to χ as

$$\begin{aligned} z_i(k+1) &= z_i(k) + \delta \quad \text{if} \quad z_i(k) < x_{w_i(k)} \\ z_i(k+1) &= z_i(k) - \delta \quad \text{if} \quad z_i(k) > x_{w_i(k)} \end{aligned} \tag{3.14}$$

for $w \in \chi$. The rules (3.14) specify $\tilde{h} : \chi \times \mathbb{R}^N \to \mathbb{R}^N$, and one can check that $\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$. In an analogous way $f : \text{perm}(N) \times \mathbb{R}^N \to \text{perm}(N)$ is extended to χ as

$$(w_i(k+1), w_{i+1}(k+1)) = (w_{i+1}(k), w_i(k)) \quad \text{if} \quad x_{w_i(k)} \geq z_{i+1}(k) \wedge x_{w_{i+1}(k)} \leq z_{i+1}(k), \tag{3.15}$$

for $w \in \chi$. The rules (3.15) model the function $\tilde{f} : \chi \times \mathbb{R}^N \to \chi$, and one can check that $\tilde{f}|_{\mathcal{U}\times\mathcal{Z}} = f$. Therefore, the system $\tilde{\Sigma} = (\tilde{f}, \tilde{h}, \chi, \mathbb{R}^N)$ is the extended system of $\Sigma = (f, h, \text{perm}(N), \mathbb{R}^N)$ (see Definition 3.3.1).

The following proposition shows that the pair $(\tilde{\Sigma}, (\chi, \leq))$ is interval compatible.

Proposition 3.4.2. *The pair $(\tilde{\Sigma}, (\chi, \leq))$, where $\Sigma = \mathcal{S}(perm(N), \mathbb{R}^N, f, h)$ is represented by the rules (3.10)–(3.11), and $(\chi, \leq)$ is given by (3.12)–(3.13), is interval compatible.*

Proof. According to Definition 3.3.5, we need to show the following two properties

(i) $\mathcal{T}_i(\tilde{\Sigma}) = [\bigwedge \mathcal{T}_i(\tilde{\Sigma}), \bigvee \mathcal{T}_i(\tilde{\Sigma})]$,

(ii) $\tilde{f} : ([\bigwedge \mathcal{T}_i(\tilde{\Sigma}), \bigvee \mathcal{T}_i(\tilde{\Sigma})]) \to [\tilde{f}(\bigwedge \mathcal{T}_i(\tilde{\Sigma})), \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}))]$ is an order isomorphism.

To simplify notation, we neglect the dependence of $\tilde{f}$ on its second argument.

Proof of (i): By (3.14) we have that $T_{(z^1,z^2)}(\tilde{\Sigma})$ is not empty if for any i we have $z_i^2 = z_i^1 + \delta$, $z_i^2 = z_i^1 - \delta$, or $z_i^2 = z_i^1$. Thus

$$T_{(z^1,z^2)}(\tilde{\Sigma}) = \begin{cases} \{w \mid x_{w_i} > z_i^1, \}, & \text{if } z_i^2 = z_i^1 + \delta \\ \{w \mid x_{w_i} < z_i^1, \}, & \text{if } z_i^2 = z_i^1 - \delta \\ \{w \mid x_{w_i} = z_i^1, \}, & \text{if } z_i^2 = z_i^1. \end{cases} \tag{3.16}$$

Because we assumed that $x_i < z_i < x_{i+1}$, we have that

$$x_{w_i} > z_i \quad \text{if and only if} \quad w_i > i$$
$$x_{w_i} < z_i \quad \text{if and only if} \quad w_i < i.$$

This, along with relations (3.16) and Definition 3.3.3, imply (i).

Proof of (ii): To show that $\tilde{f} : \mathcal{T}_i(\tilde{\Sigma}) \to [\tilde{f}(\bigwedge \mathcal{T}_i(\tilde{\Sigma})), \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}))]$ is an order isomorphism we show: a) that it is onto; and b) that it is order embedding. a) To show that it is onto, we show directly that $f(\mathcal{T}_i(\tilde{\Sigma})) = [\tilde{f}(\bigwedge \mathcal{T}_i(\tilde{\Sigma})), \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}))]$. We omit the dependence on $\tilde{\Sigma}$ to simplify notation. From the proof of (i), we deduce that the sets $\mathcal{T}_i$ are of the form $\mathcal{T}_i = (\mathcal{T}_{i,1}, ..., \mathcal{T}_{i,N})$, with $\mathcal{T}_{i,j} \in \{[1, j], [j + 1, N], [j, j]\}$. Denote by $\tilde{f}(\mathcal{T}_i)_j$ the jth coordinate set of $\tilde{f}(\mathcal{T}_i)$. By equations (3.15) we derive that $\tilde{f}(\mathcal{T}_i)_j \in \{\mathcal{T}_{i,j}, \mathcal{T}_{i,j-1}, \mathcal{T}_{i,j-1}\}$. We consider the case where $\tilde{f}(\mathcal{T}_i)_j = \mathcal{T}_{i,j-1}$; the other cases can be treated in analogous way. If $\tilde{f}(\mathcal{T}_i)_j = \mathcal{T}_{i,j-1}$ then $\tilde{f}(\mathcal{T}_i)_{j-1} = \mathcal{T}_{i,j}$. Denoting $\bigwedge \mathcal{T}_i = l$ and $\bigvee \mathcal{T}_i = u$, with $l = (l_1, ..., l_N)$ and $u = (u_1, ..., u_N)$, we have also that $\tilde{f}(l)_j = l_{j-1}$, $\tilde{f}(l)_{j-1} = l_j$, $\tilde{f}(u)_j = u_{j-1}$, $\tilde{f}(u)_{j-1} = u_j$. Thus, $\tilde{f}(\mathcal{T}_i)_j = [\tilde{f}(l)_j, \tilde{f}(u)_j]$ for all j. This in turn implies that $\tilde{f}(\mathcal{T}_i) = [\tilde{f}(l), \tilde{f}(u)]$, which is what we wanted to show. b) To show that $\tilde{f} : \mathcal{T}_i \to [\tilde{f}(\bigwedge \mathcal{T}_i), \tilde{f}(\bigvee \mathcal{T}_i)]$ is order embedding, it is enough to note again that $\tilde{f}(\mathcal{T}_i)$ is obtained by switching $\mathcal{T}_{i,j}$ with $\mathcal{T}_{i,j+1}$, $\mathcal{T}_{i,j-1}$, or leaving

it as $\mathcal{T}_{i,j}$. Therefore if $w \leq v$ for $w, v \in \mathcal{T}_i$ then $\tilde{f}(w) \leq \tilde{f}(v)$ since coordinate-wise we will compare the same numbers. By the same reasoning the reverse is also true, that is, if $\tilde{f}(w) \leq \tilde{f}(v)$ then $w \leq v$. □

The estimator $\hat{\Sigma} = (\chi \times \chi, \mathcal{Z} \times \mathcal{Z}, \chi \times \chi, (f_1, f_2), \mathrm{id})$ given in Theorem 3.3.1 can be constructed because the hypotheses of the theorem are satisfied by virtue of Proposition 3.4.1 and Proposition 3.4.2. The estimator $\hat{\Sigma}$ can be specified by the following rules

$$l_i(k+1) = i+1 \quad \text{if} \quad z_i(k+1) = z_i(k) + \delta \tag{3.17}$$

$$l_i(k+1) = 1 \quad \text{if} \quad z_i(k+1) = z_i(k) - \delta \tag{3.18}$$

$$L_{i,y}(k+1) = \max\{L_i(k), l_i(k+1)\} \tag{3.19}$$

$$\begin{aligned}(L_i(k+1), L_{i+1}(k+1)) &= (L_{i+1,y}(k+1), L_{i,y}(k+1)) \\ \text{if } x_{L_{i,y}(k+1)} \geq z_{i+1}(k) &\wedge x_{L_{i+1,y}(k+1)} \leq z_{i+1}(k)\end{aligned} \tag{3.20}$$

$$u_i(k+1) = N \quad \text{if} \quad z_i(k+1) = z_i(k) + \delta \tag{3.21}$$

$$u_i(k+1) = i \quad \text{if} \quad z_i(k+1) = z_i(k) - \delta \tag{3.22}$$

$$U_{i,y}(k+1) = \min\{U_i(k), u_i(k+1)\} \tag{3.23}$$

$$\begin{aligned}(U_i(k+1), U_{i+1}(k+1)) &= (U_{i+1,y}(k+1), U_{i,y}(k+1)) \\ \text{if } x_{U_{i,y}(k+1)} \geq z_{i+1}(k) &\wedge x_{U_{i+1,y}(k+1)} \leq z_{i+1}(k)\end{aligned} \tag{3.24}$$

initialized with $L(0) = \bigwedge \chi$ and $U(0) = \bigvee \chi$. Rules (3.17-3.18) and (3.21-3.22) take the output information z and set the lower and upper bound of $O_y(k)$, respectively. Rules (3.19) and (3.23) compute the lower and upper bound of the intersection $[L(k), U(k)] \cap O_y(k)$, respectively. Finally, rules (3.20) and (3.24) compute the lower and upper bound of the set $\tilde{f}([L(k), U(k)] \cap O_y(k))$, respectively. Also note that the rules in (3.17-3.24) are obtained

by "copying" the rules in (3.15) and correcting them by means of the output information, according to how the Kalman filter or the Luenberger observer is constructed for dynamical systems (see Kalman's seminal paper [33] or Luenberger's seminal paper [36]).

3.4.3 Complexity of the RoboFlag Drill Estimator

The amount of computation required for updating L and U according to (3.17)–(3.24) is proportional to the amount of computation required for updating the variables α in system Σ. In fact, we have $2N$ rules, $2N$ variables, and $2N$ computations of "max" and "min" of values in $\mathbb{N}$. Therefore, the computational complexity of the algorithm that generates the sequences $L(k)$ and $U(k)$ is proportional to $2N$, which is of the same order as the complexity of the algorithm that generates the α trajectories.

As established by property (iii) of Problem 3.2.1, the function of k given by $|[L(k), U(k)] \cap \mathcal{U} - \alpha(k)|$ tends to zero. This function is useful for analysis purposes, but it is not necessary to compute it at any point in the estimation algorithm proposed in equation (3.17-3.24). However, since $L(k)$ does not converge to $U(k)$ once the algorithm has converged, i.e., when $|[L(k), U(k)] \cap \mathcal{U}| = 1$, we cannot find the value of $\alpha(k)$ from the values of $U(k)$ and $L(k)$ directly. Instead of computing directly $[L(k), U(k)] \cap \mathcal{U}$, we carry out a simple algorithm, that in the case of the RoboFlag Drill example takes at most $(N^2 + N)/2$ steps and takes as inputs $L(k)$ and $U(k)$ and gives as output $\alpha(k)$ if the algorithm has converged. This is formally explained in the following paragraph.

Algorithm 3.4.1. (Refinement algorithm) Let $c_i = [L_i, U_i]$. Then the algorithm

$$(m_1, ..., m_N) = \mathit{Refine}(c_1, ..., c_N),$$

which takes assignment sets $c_1, ..., c_N$ and produces assignment sets $m_1, ..., m_N$, is such that if $m_i = \{k\}$ then $k \notin m_j$ for any $j \neq i$.

This algorithm takes as input the sets c_i and removes singletons occurring at one coordinate set from all of the other coordinate sets. By construction, it follows that $m_i \subseteq c_i$. It does this iteratively: if in the process of removing one singleton, a new one is created in

some other coordinate set, then such a singleton is also removed from all of the other coordinate sets. The refinement algorithm has two useful properties. First, the sets m_i are equal to the α_i when $[L, U] \cap \mathcal{U} = \alpha$. Second, the cardinality of the sets $m_i(k)$ is non-increasing with the time step k. These properties are proved formally in the following propositions.

Proposition 3.4.3. *If* $[L, U] \cap \mathcal{U} = \alpha$ *with* $L, U \in \chi$, *and* $c_i = [L_i, U_i]$, *then* $Refine(c_1, ..., c_N) = \alpha$.

Proof. Let c_i denote the sets $[L_i, U_i]$. Also, let $\mathcal{U}_i$ denote the set of permutations of i elements. If $[L, U] \cap \mathcal{U} = \alpha$, we note that among the sets $[L_i, U_i]$ there is at least one i for which $L_i = U_i$, and therefore we have at least one singleton to take out from all of the other coordinate sets. To show this, it is sufficient to notice that if this were not the case we would have more than one possible $\alpha \in \mathcal{U}$ in $[L(k), U(k)]$. Without loss in generality we assume that $i = N$ (if not, we can reduce to this case by performing a permutation of the coordinate sets and keeping track of the used permutation). We are left to show that the process of taking out one singleton always creates a new singleton that then needs to be removed from the other coordinate sets. Then, we remove that singleton from all of the other sets c_j for $j < N$ to obtain new sets c_j^1 whose elements take values in a set of possible $N - 1$ natural numbers. Still, there is only one $\beta \in \mathcal{U}_{N-1}$ such that $\beta \in (c_1^1, ..., c_{N-1}^1)$. Again, for this to be true there must exist j such that c_j^1, for $j \in [1, N - 1]$, is a singleton. Assume $j = N - 1$. We thus remove this singleton from all of the other sets c_j^1 for $j < N - 1$ to obtain new sets c_j^2 whose elements take values in a set of possible $N - 2$ natural numbers. Proceeding iteratively, we finally obtain $m_1 = c_1^{N-1}$,...,$m_{N-1} = c_{N-1}^1$, $m_N = c_N$, which implies that the m_i are singletons. Since $\alpha_i \in m_i$ by construction, we have proved what we wanted. □

Proposition 3.4.4. *Let* $c_i(k) = [L_i(k), U_i(k)]$, *and denote by* $m_i(k)$ *the sets obtained with the refinement algorithm. Then*

$$\sum_{i=1}^{N} |m_i(k+1)| \leq \sum_{i=1}^{N} |m_i(k)|.$$

Proof. Let us denote the variables at step $k + 1$ with primed variables and the variables at step k with unprimed variables. The proof proceeds by showing that for each j there exists

a k such that $m'_j \subseteq m_k$. By equations (3.17-3.24) we deduce that we can have one of the following cases for each i: (a) $c'_i \subseteq c_{i+1} \ \wedge \ c'_{i+1} \subseteq c_i$, (b) $c'_i \subseteq c_i$, (c) $c'_i \subseteq c_{i-1} \ \wedge \ c'_{i-1} \subseteq c_i$. Let us consider case (a), the other cases can be treated in an analogous way. Let c_j be a singleton. In the refinement process it is deleted from any other set, so that we have $c_i = m_i + c_j$ for all i. Assume that in the first singleton removal process no new singletons are created. We have one of the following situations: $c'_j \subseteq c_{j+1} \ \wedge \ c_{j+1} \subseteq c_j$, $c'_j \subseteq c_j$, $c'_j \subseteq c_{j-1} \ \wedge \ c'_{j-1} \subseteq c_j$. This implies that one of the c'_k is equal to the singleton c_j. The sets m'_i are created removing such singleton for all the other sets, so that we obtain $m'_i + c_j = c'_i \subseteq c_{i+1} = m_{i+1} + c_j$ and $m'_{i+1} + c_j = c'_{i+1} \subseteq c_i = m_i + c_j$. This in turn implies that $m'_i \subseteq m_{i+1}$ and $m'_{i+1} \subseteq m_i$. Because this holds for any i, we have that $\sum_{i=1}^{N} |m'_i| \leq \sum_{i=1}^{N} |m_i|$. This reasoning can be generalized to the case where a singleton removal process creates new singletons. □

As a final remark, note that the sets m_i are not used in the update laws of the estimation algorithm, they are just computed at each step from the set $[L, U]$ in order to extract α from it when the algorithm has converged.

3.4.4 Simulation Results

The RoboFlag Drill system represented in the rules (3.10) and (3.11) has been implemented in Matlab together with the estimator reported in the rules (3.17-3.24). Figure 3.7 (left) shows the behavior of the quantities

$$V(k) = |[L(k), U(k)] \cap \mathcal{U}|$$

and

$$E(k) = \frac{1}{N} \sum_{i=1}^{N} |\alpha_i(k) - i|.$$

$V(k)$ represents the cardinality of the set of all possible assignments at each step. This quantity gives an idea of the convergence rate of the estimator. $E(k)$ is a function of α, and it is not increasing along the executions of the system $\Sigma = \mathcal{S}(\text{perm}(N), \mathbb{R}^N, f, h)$. This quantity is showing the rate of convergence of the α assignment to its equilibrium $(1, ..., N)$.

In Figure 3.7 (right) we show the results for $N = 30$ robots per team. In particular, we report

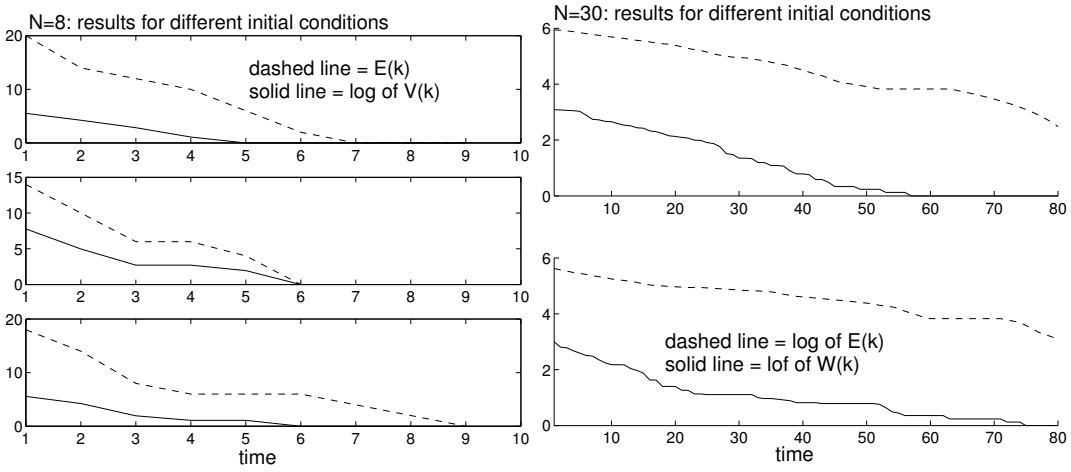


Figure 3.7: (Left) Example with N=8: note that the function $V(k)$ is always non-increasing because the set $\chi - \mathcal{U}$ is invariant under $\tilde{f}$. (Right) Example with N=30: note that the function $W(k)$ is always non-increasing, and its logarithm is converging to zero.

the logarithm of $E(k)$ and the logarithm of $W(k)$ defined as

$$W(k) = \frac{1}{N} \sum_{i=1}^{N} |m_i(k)|,$$

which by virtue of Proposition 3.4.3 and Proposition 3.4.4 is non-increasing and converging to one, that is, the sets $(m_1(k), ..., m_N(k))$ converge to $\alpha(k) = (\alpha_1(k), ..., \alpha_N(k))$. In the same figure, we notice that when $W(k)$ converges to one, $E(k)$ has not converged to zero yet. This shows that the estimator is faster than the dynamics of the system under study.

In this chapter, we have proposed an estimator $\hat{\Sigma} = (\chi \times \chi, \mathcal{Z} \times \mathcal{Z}, \chi \times \chi, (f_1, f_2), \mathrm{id})$ on a lattice $(\chi, \leq)$ for a DTS $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ with $\mathcal{U} \subseteq \chi$. Such an estimator can be constructed if the extended system $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ is such that the pair $(\tilde{\Sigma}, (\chi, \leq))$ is interval compatible. In the next chapter, we investigate when the pair $(\tilde{\Sigma}, (\chi, \leq))$ is interval compatible, and what possible causes can be of the estimator complexity.

Chapter 4

Existence of Discrete State Estimators on a Lattice

In the previous chapter, we have shown that if a system can be extended to a lattice in a way such that the system extension and the lattice are order compatible, the estimator on a lattice given in Theorem 3.3.1 can be implemented. In this chapter, we give a characterization of what observable means in terms of the extensibility of a system into an extended system that is interval compatible with a lattice $(\chi, \leq)$. We show that if the system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ is observable, there always exists a lattice $(\chi, \leq)$ such that the pair $(\tilde{\Sigma}, (\chi, \leq))$ is interval compatible. The size of the set χ is singled out as a cause of complexity, and a worst case size is computed. In particular, the worst case size of the lattice never exceeds the size of the observer tree proposed in [13]. As a consequence, the method proposed in this thesis is in the worst case computationally equivalent to previously proposed methods. For systems where $\mathcal{U}$ can be immersed in a space equipped with algebraic properties, as is the case for the RoboFlag Drill, a preferred lattice structure $(\chi, \leq)$ exists where joins and meets can be efficiently computed and represented by exploiting the algebra. For these systems, the estimation methodology proposed in this thesis reduces complexity with respect to enumeration methods. However, the pair $(\tilde{\Sigma}, (\chi, \leq))$ is not necessarily interval compatible for any $(\chi, \leq)$. We propose a way of constructing the estimator on a chosen lattice by finding a generalization of Σ for which there exists an extension on the lattice $(\chi, \leq)$ with the desired properties. A possible way of determining a generalization of Σ is by creating a nondeterministic approximation of the system. This is explained in the

following chapter. The results of this chapter have appeared in [26].

Note that, as in the case of the RoboFlag Drill, there are a number of applications in which the discrete state dynamics evolves naturally on partially ordered sets. Resource allocation problems involving moving resources (agents), as in the case of air traffic controlled systems ([6, 42]) or weapon-target assignment problems, are examples where the tasks are usually associated with positions in Euclidean space, where the usual cone partial order induces a partial order on the tasks. In the case of dynamic scheduling for distributed computing, the set of processes that need to be allocated to resources is typically partially ordered according to priorities [10], and the allocation to resources is dynamically established on the basis of such partial ordering. In addition, there are plenty of systems where a partial order among events is naturally established by a causal order relation, as in the case of message passing based distributed systems [45], or in the case of human activities [27]. An example from this class, is presented in Chapter 7. Also, all discrete event systems implemented as Petri nets [16] are characterized by order preserving state transition functions with respect to a suitable partial order. In Chapter 7, we show how our algorithms apply to this case. Most of these examples are also distributed, meaning that the size of the discrete state is so large as to render prohibitive the estimation problem if the partial order is not explicitly taken into account in the estimator design.

This chapter is organized in two sections. In Section 4.1, the existence result for the estimator on a lattice is given. In Section 4.2, a possible way for constructing the estimator on a chosen lattice is given.

4.1 Estimator Existence

Consider the deterministic transition system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$. In order to show the link between the original system and its extension, it is useful to define the Σ-transition sets and the Σ-transition classes as defined for the extended system $\tilde{\Sigma} = \mathcal{S}(\mathcal{U}, \mathcal{Z}, \tilde{f}, \tilde{h})$ in Definition 3.3.2 and Definition 3.3.3, respectively.

Definition 4.1.1. The non-empty sets $T_{(z^1,z^2)}(\Sigma) = \{\alpha \in \mathcal{U} \mid z^2 = h(\alpha, z^1)\}$, for $z^1, z^2 \in \mathcal{Z}$, are named the Σ-*transition sets*.

Definition 4.1.2. The set $\mathcal{T}(\Sigma) = \{\mathcal{T}_1(\Sigma), ..., \mathcal{T}_M(\Sigma)\}$, with $\mathcal{T}_i(\Sigma)$ such that

(i) for any $\mathcal{T}_i(\Sigma) \in \mathcal{T}(\Sigma)$ there is $(z^1, z^2) \in \mathcal{Z}$ such that $\mathcal{T}_i(\Sigma) = T_{(z^1,z^2)}(\Sigma)$;

(ii) for any $z^1, z^2 \in \mathcal{Z}$ for which $T_{(z^1,z^2)}(\Sigma)$ is not empty, there is $j \in \{1, ..., M\}$ such that $T_{(z^1,z^2)}(\Sigma) = \mathcal{T}_j$;

is the *set of* Σ*-transition classes*.

Note that the set $\mathcal{T}(\Sigma)$ is finite as we assumed at the beginning that the set $\mathcal{U}$ is finite. Each Σ-transition set $T_{(z_1,z^2)}(\Sigma)$ contains all of α values in $\mathcal{U}$ that allow the transition from z^1 to z^2 through the function h. Note also that for any $z^1, z^2 \in \mathcal{Z}$ we have $T_{(z^1,z^2)}(\Sigma) \subseteq T_{(z^1,z^2)}(\tilde{\Sigma})$ because $\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$ and $\mathcal{U} \subseteq \chi$. This in turn implies that $\mathcal{T}_i(\Sigma) \subseteq \mathcal{T}_i(\tilde{\Sigma})$.

We also assume that all of the executions contained in the ω^+-limit set of Σ, $\omega(\Sigma)$, are distinguishable. More formally we have:

Assumption 4.1.1. The ω^+-limit set of $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$, $\omega(\Sigma)$, is such that for any two different executions σ_1, σ_2 with $\sigma_1(0), \sigma_2(0) \in \omega(\Sigma)$ there is $k \in \mathbb{N}$ such that $\sigma_1(k)(z) \neq \sigma_2(k)(z)$.

In the case in which ω^+-limit set is just one fixed point, this assumption is always trivially verified. In the case where ω^+-limit set is made up by fixed points and limit cycles, the assumption requires that any two different fixed points have different output values, otherwise two different executions starting in the two fixed points will not be distinguishable.

Lemma 4.1.1. *Consider the deterministic transition system* $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$*. Let* $\omega(\Sigma)$ *verify Assumption 4.1.1. Then,* Σ *is observable if and only if* $f : (\mathcal{T}_j(\Sigma), z) \to f(\mathcal{T}_j(\Sigma), z)$ *is one to one for any* $j \in \{1, .., M\}$ *and for any* $z \in \mathcal{Z}$*.*

Proof. ($\Longrightarrow$). Let us show that if the system is observable then for any $z \in \mathcal{Z}$ and any $j \in \{1, .., M\}$ we have that $f : (\mathcal{T}_j(\Sigma), z) \to f(\mathcal{T}_j(\Sigma), z)$ is one to one. We have to show that if $\alpha_a \neq \alpha_b$ and $\alpha_a, \alpha_b \in \mathcal{T}_j(\Sigma)$ for some j, then $f(\alpha_a, z) \neq f(\alpha_b, z)$. Suppose instead that $f(\alpha_a, z) = f(\alpha_b, z)$, this means that the two executions σ_a, σ_b starting at $\sigma_a(0) = (\alpha_a, z)$ and $\sigma_b(0) = (\alpha_b, z)$ have the same output sequence, but they are different. This means that

they are not distinguishable, and therefore the system is not observable. This contradicts the assumption.

($\Longleftarrow$). We assume that for any $z \in \mathcal{Z}$ we have that $f : (\mathcal{T}_j(\Sigma), z) \to f(\mathcal{T}_j(\Sigma), z)$ is one to one, and that Assumption 4.1.1 is verified. We need to show that for any $\sigma_1 \neq \sigma_2$ there is $k \in \mathbb{N}$ such that $g(\sigma_1(k)) \neq g(\sigma_2(k))$, that is, σ_1 and σ_2 are distinguishable. Let then σ_1 and σ_2 be two executions such that $\sigma_1(0) \neq \sigma_2(0)$. Assume that $g(\sigma_1) = g(\sigma_2)$. This implies that $\sigma_1(k) = (\alpha_1(k), z(k))$ and $\sigma_2(k) = (\alpha_2(k), z(k))$. This implies that $\alpha_1(k) \neq \alpha_2(k)$ for all k, because $\alpha_1(k)$ and $\alpha_2(k)$ are in $T_{(z(k+1),z(k))}$ (which coincides with a $\mathcal{T}_i$ for some i by Definition 4.1.2), and we assumed that if $\alpha_1(k) \neq \alpha_2(k)$ then $f(\alpha_1(k), z(k)) \neq f(\alpha_2(k), z(k))$. This in turn implies that for $k \geq k_{\sigma_1}$ and $k \geq k_{\sigma_2}$, $\sigma_1(k) \in \omega(f, h)$, $\sigma_2(k) \in \omega(\Sigma)$, $\sigma_1(k) \neq \sigma_2(k)$ and $g(\sigma_1) = g(\sigma_2)$. This contradicts the assumption. Therefore if $\sigma_1(0) \neq \sigma_2(0)$, we have that $g(\sigma_1) \neq g(\sigma_2)$, which implies that σ_1 and σ_2 are distinguishable and by Definition 2.2.4 implies that Σ is observable with output z. □

This lemma shows that observability can be determined by checking if the function f is one to one on the Σ-transition classes $\mathcal{T}_j(\Sigma)$, provided that the executions evolving in $\omega(\Sigma)$ are distinguishable. This lemma is used in the following theorem, which gives an alternative characterization of what observable means in terms of extensibility of the system Σ into a system $\tilde{\Sigma}$ that is, interval compatible with a lattice $(\chi, \leq)$.

Theorem 4.1.1. *(Observability on bounded lattices) Consider the deterministic transition system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$. Let $\omega(\Sigma)$ verify Assumption 4.1.1. Then the following are equivalent:*

(i) System Σ is observable;

(ii) There exists a complete lattice $(\chi, \leq)$ with $\mathcal{U} \subseteq \chi$, such that the extension $\tilde{\Sigma} = (\tilde{f}, \tilde{h}, \chi, \mathcal{Z})$ of Σ on χ is such that $(\tilde{\Sigma}, (\chi, \leq))$ is interval compatible.

Proof. ((i) $\Rightarrow$ (ii)) We show the existence of a lattice $(\chi, \leq)$ and of an extended system $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ with $(\tilde{\Sigma}, (\chi, \leq))$ an interval compatible pair by construction. Define $\chi := \mathcal{P}(\mathcal{U})$, and $(\chi, \leq) := (\mathcal{P}(\mathcal{U}), \subseteq)$.

To define $\tilde{h}$, define the sublattices $(\mathcal{T}_i(\tilde{\Sigma}), \leq)$ of $(\chi, \leq)$ for $i \in \{1, ..., M\}$, by $(\mathcal{T}_i(\tilde{\Sigma}), \leq) := (\mathcal{P}(\mathcal{T}_i(\Sigma)), \subseteq)$ as shown in Figure 4.1. As a consequence, for any given $z^1, z^2 \in \mathcal{Z}$ such that $z^2 = h(\alpha, z^1)$ for $\alpha \in \mathcal{T}_i(\Sigma)$ for some i, we define $z^2 = \tilde{h}(w, z^1)$ for any $w \in \mathcal{T}_i(\tilde{\Sigma})$. Clearly, $\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$, and $\mathcal{T}_i(\tilde{\Sigma})$ for any i is an interval sublattice of the form $\mathcal{T}_i(\tilde{\Sigma}) = [\bot, \bigvee \mathcal{T}_i(\tilde{\Sigma})]$.

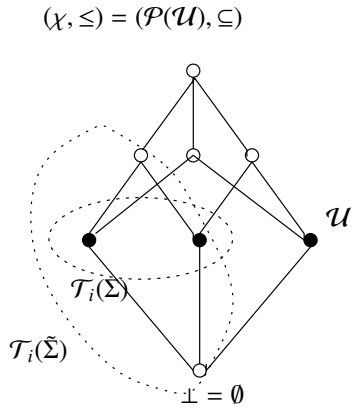


Figure 4.1: Example of the Σ and $\tilde{\Sigma}$ transition classes with $\mathcal{U}$ (dark elements) composed by three elements.

The function $\tilde{f}$ is defined in the following way. For any $x, w \in \chi$ and $\alpha \in \mathcal{U}$ we have

$$\begin{cases} \tilde{f}(x \vee w) & := \tilde{f}(x) \vee \tilde{f}(w) \\ \tilde{f}(x \wedge w) & := \tilde{f}(x) \wedge \tilde{f}(w) \\ \tilde{f}(\bot) & := \bot \\ \tilde{f}(\alpha) & := f(\alpha), \end{cases} \tag{4.1}$$

where we have omitted the dependency on the z variable (that is, kept fixed) to simplify notation. We prove first that $\tilde{f} : \mathcal{T}_i(\tilde{\Sigma}) \to [\bot, \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}))]$ is onto. We have to show that for any $w \in [\bot, \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}))]$, with $w \neq \bot$, there is $x \in [\bot, \bigvee \mathcal{T}_i(\tilde{\Sigma})]$ such that $w = \tilde{f}(x)$. Since $\bigvee \mathcal{T}_i(\tilde{\Sigma}) = \alpha_1 \vee ... \vee \alpha_p$ for $\{\alpha_1, ..., \alpha_p\} = \mathcal{T}_i(\Sigma)$, we have also that $\tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma})) = f(\alpha_1) \vee .... \vee f(\alpha_p)$ by virtue of equations (4.1). Because $w \leq \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}))$, we have that $w = f(\alpha_{j_1}) \vee ... \vee f(\alpha_{j_m})$ for $j_k \in \{1, ..., p\}$ and $m < p$. This in turn implies, by equations (4.1), that $w = \tilde{f}(\alpha_{j_1} \vee ... \vee \alpha_{j_m})$. Since $x := \alpha_{j_1} \vee ... \vee \alpha_{j_m} < \bigvee \mathcal{T}_i(\tilde{\Sigma})$, we have proved that $w = \tilde{f}(x)$ for $x \in \mathcal{T}_i(\tilde{\Sigma})$. Second, we notice that $\tilde{f} : \mathcal{T}_i(\tilde{\Sigma}) \to [\bot, \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}))]$ is one to

one because of Lemma 4.1.1. Thus, we have proved that $\tilde{f} : \mathcal{T}_i(\tilde{\Sigma}) \to [\perp, \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}))]$ is a bijection, and by equations (4.1) it is also a homomorphism. We then apply Proposition 2.1.1 to obtain the result.

((ii) $\Rightarrow$ (i)). To show that (ii) implies that $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ is observable, we apply Lemma 4.1.1. In particular, $(\tilde{\Sigma}, (\chi, \leq))$ being interval compatible implies that $\tilde{f} : \mathcal{T}_i(\tilde{\Sigma}) \to [\tilde{f}(\bigwedge \mathcal{T}_i(\tilde{\Sigma})), \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}))]$ is one to one for any i. This, along with Assumption 4.1.1, by Lemma 4.1.1 implies that the system is observable. □

This result links the property of a pair $(\tilde{\Sigma}, (\chi, \leq))$ being interval compatible with the observability properties of the original system Σ.

Theorem 4.1.1 shows that an observable system admits a lattice and a system extension that satisfy interval compatibility by constructing them, in a way similar to the way one shows that a stable dynamical system has a Lyapunov function. However, the lattice constructed in the proof of the theorem is impractical for the implementation of the estimator of Theorem 3.3.1 when the size of $\mathcal{U}$ is large because the size of the representation of the elements of χ is large as well. However, one does not need to have $\chi = \mathcal{P}(\mathcal{U})$, but it is enough to have in χ the elements that the estimator needs, that is, the elements in the $\tilde{\Sigma}$-transition classes. With this consideration, the following proposition computes the worst case size of χ.

Proposition 4.1.2. *Consider the system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$, with $f : \mathcal{U} \to \mathcal{U}$. Assume that the sets $\{\mathcal{T}_1(\Sigma), ..., \mathcal{T}_m(\Sigma)\}$ are all disjoint. Then there exist an extension $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ with $|\chi| \leq 2|\mathcal{U}|^2$.*

Proof. We construct the worst case $(\chi, \leq)$ by adding in it all the elements that the estimator in Theorem 3.3.1 needs. These are in the set of subsets of $\mathcal{U}$ ordered according to inclusion. Let $\mathcal{T}_i(\Sigma)$ be a Σ-transition class. For simplifying notation, we omit the dependence on Σ denoting $\mathcal{T}_i(\Sigma)$ by $\mathcal{T}_i$. The proof proceeds in two steps: 1) we show that for any $\mathcal{T}_i$, the last element of the sequence $\{\mathcal{T}_i, f(\mathcal{T}_i) \cap \mathcal{T}_{i_1}, f(f(\mathcal{T}_i) \cap \mathcal{T}_{i_1}) \cap \mathcal{T}_{i_2}, ..., f(f(...f(\mathcal{T}_i) \cap \mathcal{T}_{i_1}...)) \cap \mathcal{T}_{i_n}\}$ is a singleton for $n < |\mathcal{U}|$, for any $i_j \in \{1, ..., m\}$; 2) the *jth* element of the above sequence can generate at most $|\mathcal{U}|$ nonempty sets for any combination $(i_1, ..., i_{j-1})$ and for any j.

Proof of 1). Let ω_α denote the ω^+-limit set of f. Since all of the executions are converging to the ω^+-limit set, it is enough to show that any two executions starting in the ω_α, will distinguish from each other in less than $n = |\omega_\alpha|$. We proceed by contradiction. Define the function $Y : \mathcal{U} \to \{\mathcal{Y}_1, ..., \mathcal{Y}_m, \}$ such that $Y(\alpha) = \mathcal{Y}_i$ if $\alpha \in \mathcal{T}_i$. Assume that there are $x_i, x_j \in \omega_\alpha$ such that

$$(a)\ Y(f^k(x_i)) = Y(f^k(x_j)) \text{ for any } k < n \text{ and}$$

$$(b)\ Y(f^n(x_i)) \neq Y(f^n(x_j)).$$

Since x_i and x_j belong each to a limit cycle, there are k_j, k_i such that $f^{n-k_i}(x_i) = f^n(x_i)$ and $f^{n-k_j}(x_j) = f^n(x_j)$. As a consequence, we have by (b) that

$$(c)\ Y(f^{n-k_i}(x_i)) \neq Y(f^{n-k_j}(x_j)).$$

Assume without loss in generality that $k_i \geq k_j$. If x_i and x_j belong to the same limit cycle, we have $k_i = k_j$, and therefore we contradict (a). If $k_i > k_j$, x_i and x_j belong to different limit cycles, and k_i and k_j are the respective limit cycle lengths. Thus $k_i + k_j \leq n$. Thus, by virtue of (a) we have $Y(f^{n-(k_i+k_j)}(x_i)) = Y(f^{n-(k_i+k_j)}(x_j))$ and by the periodicity of trajectories in the limit cycles, we have $f^{n-(k_i+k_j)}(x_i) = f^{n-k_j}(x_i)$ and $f^{n-(k_i+k_j)}(x_j) = f^{n-k_i}(x_j)$. As a consequence, $Y(f^{n-(k_i+k_j)}(x_i)) = Y(f^{n-k_j}(x_i))$ and $Y(f^{n-(k_i+k_j)}(x_j)) = Y(f^{n-k_i}(x_j))$. By (a), we also have that $Y(f^{n-k_i}(x_j)) = Y(f^{n-k_i}(x_i))$ and $Y(f^{n-k_j}(x_i)) = Y(f^{n-k_j}(x_j))$. One can verify that the set of these relations are inconsistent with (c).

Proof of 2). Since the $\mathcal{T}_i$s are all disjoint, the *jth* element of the sequence in 2) can have at most $|\mathcal{T}_i|$ nonempty intersections for any combination of $(i_1, ..., i_{j-1})$ for any j. Then for any j, we can have at most $\sum_i |\mathcal{T}_i| = |\mathcal{U}|$ nonempty intersections.

Since the estimator needs all of these $|\mathcal{U}|^2$ elements and the "f" of these elements, the size of $(\chi, \leq)$ is at most $2|\mathcal{U}|^2$. □

Example In this example, we show what the lattice $(\chi, \leq)$ looks like in the case of an automaton driving the discrete state dynamics. Consider the automaton reported in Figure 4.2 where $\mathcal{U} = \{\alpha_1, .., \alpha_5\}$, and $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2\}$. The lattice $(\chi, \leq)$ can be constructed by

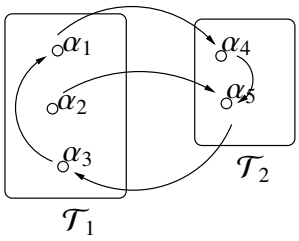


Figure 4.2: Automaton example.

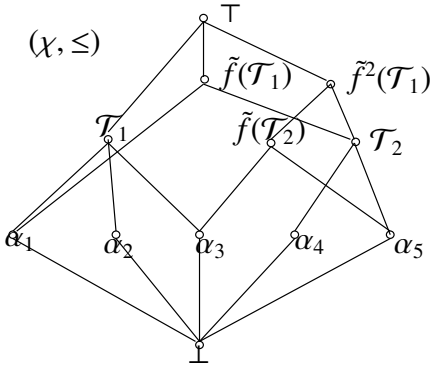


Figure 4.3: Automaton example: lattice $(\chi, \leq)$.

following the procedure in the proof of Proposition 4.1.2, and it is shown in Figure 4.3. The way it is constructed is as follows. For each set $\mathcal{T}_i$, include an element that represents the set itself (denoted $\mathcal{T}_i$ in the diagram), and add edges going down to the elements it contains. Then, include an element that represents the set $f(\mathcal{T}_i)$, denoted $\tilde{f}(\mathcal{T}_i)$ in the diagram, and add edges representing the inclusion relation accordingly. Then, include a set of elements representing each the intersection of $f(\mathcal{T}_i)$ with the sets $\mathcal{T}_i$, with the edges representing the various inclusion relations accordingly. One proceeds this way until the intersection sets are singletons (α_i). At such a point, no new element needs to be added.

□

The size of χ gives an idea of how many values of joins and meets need to be stored. In the case of the RoboFlag example with $N = 4$ robots per team, the size of $\mathcal{P}(\mathcal{U})$ is 16778238, while the worst case size given in Proposition 4.1.2 is 576, and the size of the lattice χ proposed in Section 3.4.2 is $4^4 = 256$. Thus the estimate given by Proposition 4.1.2 significantly reduces the size of χ given by $\mathcal{P}(\mathcal{U})$. Note that the size of the lattice

proposed in Section 3.4.2 is smaller than 576 because there are pairs of elements that have the same join, for example, the pairs $(3,1,4,2),(4,2,1,3)$, and $(4,2,1,3),(2,1,4,3)$ have the same join, that is, $(4,2,4,3)$.

This proposition shows that the worst case computation needed for implementing our estimator is the same as the one needed in Caines [13], where the observer tree method is proposed. The main advantage of the method proposed in this thesis is clear when the space of discrete variables can be immersed in a lattice whose order relations can be computed algebraically ($(\chi,\leq)$ does not need to be stored). In such a case, one needs to find a better lattice, if it exists, considering its size, the representation of its elements, and the complexity of computing joins and meets. In the RoboFlag Drill, for example, such a better lattice exists. Even if the size of $\mathcal{U}$ is $N!$ (which is huge if N is large) the lattice $(\chi,\leq)$ is such that its elements can be represented by a set of N natural numbers plus joins and meets, and $\tilde{f}$ can be computed by using the algebra naturally associated with $\mathbb{N}^N$. Thus, some systems have a preferred lattice structure that drastically reduces complexity. For these systems however, the extended system and such a preferred lattice structure are not always interval compatible. In such a case, we propose in the following section a way to construct an estimator on the desired lattice even if the interval compatibility condition is not satisfied. The trade-off is that the convergence speed of the estimator can be lower.

4.2 Existence of an Estimator on a Chosen Lattice

In this section, we consider the case in which there is a preferred lattice structure $(\chi,\leq)$ in which the order relations can be computed algebraically, but there is no system extension $\tilde{\Sigma}$ such that the pair $(\tilde{\Sigma},(\chi,\leq))$ is interval compatible. We thus look for an over-approximation of the system Σ that might be interval compatible with the desired lattice $(\chi,\leq)$. Such an over-approximation is called a weakly equivalent generalization and is defined in term of the set of all executions $\mathcal{E}(\Sigma)$.

Definition 4.2.1. Consider the deterministic transition system $\Sigma = \mathcal{S}(\mathcal{U},\mathcal{Z},f,h)$. We define $\Sigma_{\geq} = \mathcal{S}(\mathcal{U}_{\geq},\mathcal{Z},f_{\geq},h)$ to be a Σ-*weakly equivalent generalization* of Σ on $\mathcal{U}_{\geq}$ with $\mathcal{U}\subseteq\mathcal{U}_{\geq}$ if

(i) $\mathcal{E}(\Sigma) \subseteq \mathcal{E}(\Sigma_{\geq})$;

(ii) any $\sigma_{\Sigma_{\geq}} \in \mathcal{E}(\Sigma_{\geq})$ such that $\{\sigma_{\Sigma_{\geq}}(k)(z)\}_{k\in\mathbb{N}} = \{\sigma_{\Sigma}(k)(z)\}_{k\in\mathbb{N}}$, for some execution $\sigma_{\Sigma} \in \mathcal{E}(\Sigma)$, is such that $\sigma_{\Sigma_{\geq}} \sim \sigma_{\Sigma}$.

Item (i) establishes that $\Sigma_{\geq}$ is a generalization of Σ, denoted $\Sigma \subseteq \Sigma_{\geq}$. Moreover, (ii) establishes that those executions of $\Sigma_{\geq}$ that have the same output sequence as one of the executions, σ_{Σ}, of Σ are equivalent to σ_{Σ}. As a consequence, if the system Σ is observable (or weakly observable), its Σ-weakly equivalent generalization $\Sigma_{\geq}$ is weakly observable on the set of executions of Σ. For weakly observable systems, Theorem 3.3.1 can be applied by substituting the assumption of the pair $(\tilde{\Sigma}, (\chi, \leq))$ being interval compatible with a weaker assumption that we call *weak interval compatibility* defined as follows.

Definition 4.2.2. (Weak interval compatibility) Given the extended system $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ of $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ on $(\chi, \leq)$. The pair $(\tilde{\Sigma}, (\chi, \leq))$ is said to be *weakly interval compatible* if

(i) each $\tilde{\Sigma}$-transition class, $\mathcal{T}_i(\tilde{\Sigma}) \in \mathcal{T}(\tilde{\Sigma})$, is an interval sublattice of $(\chi, \leq)$:

$$\mathcal{T}_i(\tilde{\Sigma}) = [\bigwedge \mathcal{T}_i(\tilde{\Sigma}), \bigvee \mathcal{T}_i(\tilde{\Sigma})];$$

(ii) $\tilde{f} : ([L, U], z) \longrightarrow [\tilde{f}(L, z), \tilde{f}(U, z)]$ is order preserving for any $[L, U] \subseteq \mathcal{T}_i(\tilde{\Sigma})$, and any $z \in \mathcal{Z}$ and for any $i \in \{1, ..., M\}$;

(iii) $\tilde{f} : ([L, U], z) \longrightarrow [\tilde{f}(L, z), \tilde{f}(U, z)]$ is onto for any $[L, U] \subseteq \mathcal{T}_i(\tilde{\Sigma})$ for any $z \in \mathcal{Z}$ and for any $i \in \{1, ..., M\}$.

We have this difference between observable systems and weakly observable systems because in a weakly observable system, two executions sharing the same output can collapse one onto the other, thus there cannot be any extension $\tilde{f}$ that is a bijection between the output lattice and the set it is mapped to. Thus, we can restate Theorem 3.3.1 for weakly observable systems in the following way.

Theorem 4.2.1. *Assume that the deterministic transition system* $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ *is weakly observable. If there is a lattice* $(\chi, \leq)$*, such that the pair* $(\tilde{\Sigma}, (\chi, \leq))$ *is weakly interval compatible, then the deterministic transition system with input* $\hat{\Sigma} = (\chi \times \chi, \mathcal{Z} \times \mathcal{Z}, \chi \times \chi, (f_1, f_2), id)$ *with*

$$\begin{aligned} f_1(L(k), y(k), y(k+1)) &= \tilde{f}(L(k) \vee \bigwedge O_y(k), y(k)) \\ f_2(U(k), y(k), y(k+1)) &= \tilde{f}(U(k) \wedge \bigvee O_y(k), y(k)) \end{aligned}$$

solves the discrete state estimation problem stated in Problem 3.2.1.

If we can find a Σ-weakly equivalent generalization $\Sigma_{\geq}$ for Σ that is weakly interval compatible with the desired lattice χ, we can construct the estimator for the system Σ by using $\Sigma_{\geq}$. This is formally stated in the following proposition.

Proposition 4.2.2. *If the system* $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ *is observable (or weakly observable) and its* Σ*-weakly equivalent generalization* $\Sigma_{\geq} = \mathcal{S}(\mathcal{U}_{\geq}, \mathcal{Z}, f_{\geq}, h)$ *is such that the pair* $(\tilde{\Sigma}_{\geq}, (\chi, \leq))$ *is weakly interval compatible for a given* $(\chi, \leq)$ *and* $\mathcal{U}_{\geq} \subseteq \chi$*, then Theorem 4.2.1 can be applied to* $\Sigma_{\geq}$ *with* $\alpha(k) = \sigma_{\Sigma}(k)(\alpha)$ *and* $z(k) = \sigma_{\Sigma}(k)(z)$*.*

This way, we construct the estimator using $f_{\geq}$, but we estimate the value of α corresponding to the execution of Σ whose output z we are measuring. The proof of this proposition can be carried out easily by using directly (i) and (ii) of Definition 4.2.1. The counterpart is that if the Σ-weakly equivalent generalization is too rough an over-approximation of Σ, the convergence speed can be low.

A way of constructing a Σ-weakly equivalent generalization of Σ is to find a nondeterministic function $f_{\geq} : \mathcal{U} \times \mathcal{Z} \to \mathcal{P}(\mathcal{U})$ such that if $\alpha(k) = \sigma_{\Sigma}(k)(\alpha)$ and $z(k) = \sigma_{\Sigma}(k)(z)$, then $\alpha(k+1) \in f_{\geq}(\alpha(k), z(k))$. The function $f_{\geq}$ maps an element to a set of possible values in $\mathcal{U}$, and $\mathcal{U}_{\geq} = \mathcal{U}$. We show in the following chapter how the results obtained for deterministic systems carry to nondeterministic systems.

Chapter 5

Discrete State Estimators on a Lattice for Nondeterministic Systems

In this chapter, we outline the basic ideas that allow us to generalize the results of Chapters 3 and 4 to nondeterministic transition systems. The systems considered in this chapter are not probabilistic but rather nondeterminism arises because a state is updated to a known set of possible values instead of being updated to one value only. This is a way of taking modeling uncertainty into account. Such uncertainty can be due to poor knowledge of the dynamics, or to timing uncertainties that often happen in concurrent systems. In these systems, the state of the agents can be updated in an asynchronous fashion, and the order in which each state of each agent is updated is not known *a priori*. In Section 5.1, basic definitions for nondeterministic transition systems are given. In Section 5.2, the estimator proposed in the previous chapters is constructed and the existence result proved. The chapter is concluded with a nondeterministic example in Section 5.3. The results of this chapter appeared in [24].

5.1 Basic Definitions

Definition 5.1.1. (Nondeterministic transition systems) A *nondeterministic transition system* (NTS) is the tuple $\Sigma = (S, \mathcal{Y}, F, g)$, where

(i) S is a set of states with $s \in S$;

(ii) $\mathcal{Y}$ is a set of outputs with $y \in \mathcal{Y}$;

(iii) $F : S \to \mathcal{P}(S)$ is the state transition set-valued function;

(iv) $g : S \to \mathcal{Y}$ is the output function.

An execution of Σ is any sequence $\sigma = \{s(k)\}_{k\in\mathbb{N}}$ such that $s(0) \in S$ and $s(k+1) \in F(s(k))$ for all $k \in \mathbb{N}$. As opposed to a deterministic transition systems, in an nondeterministic transition system F maps an element to a set, and thus it is a set-valued function. The Definitions 2.2.2, 2.2.5, and 2.2.6, which are related to the weak observability property, can be rewritten the same way for NTSs by replacing "deterministic transition system" with "nondeterministic transition system" and by taking into account that F is a set-valued map. As done for deterministic transition systems, we consider nondeterministic transition systems with the special structure

(i) $S = \mathcal{U} \times \mathcal{Z}$ with $\mathcal{U}$ a finite set and $\mathcal{Z}$ a finite dimensional space;

(ii) $F = (f, h)$, where $f : \mathcal{U} \times \mathcal{Z} \to \mathcal{P}(\mathcal{U})$ and $h : \mathcal{U} \times \mathcal{Z} \to \mathcal{Z}$;

(iii) $g(\alpha, z) := z$, where $\alpha \in \mathcal{U}$, $z \in \mathcal{Z}$, and $\mathcal{Y} = \mathcal{Z}$.

We denote this class of nondeterministic transition systems by $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$, and we associate to the tuple $(\mathcal{U}, \mathcal{Z}, f, h)$ the equations

$$\begin{aligned} \alpha(k+1) &\in f(\alpha(k), z(k)) \\ z(k+1) &= h(\alpha(k), z(k)) \\ y(k) &= z(k), \end{aligned} \tag{5.1}$$

if f is a set-valued map. Given a lattice $(\chi, \leq)$ with $\mathcal{U} \subset \chi$, the extension $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ of Σ is defined in a way similar to the way it is defined for deterministic transition systems (see Definition 3.3.1), but in this case $\tilde{\Sigma}$ is nondeterministic itself and $\mathcal{U}$ is not required to be invariant under $\tilde{f}$.

Definition 5.1.2. Given the nondeterministic transition system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$, a *N-extension of* Σ *on* χ, with $\mathcal{U} \subseteq \chi$ and $(\chi, \leq)$ a complete lattice, is any system $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$, such that

(i) $\tilde{f} : \chi \times \mathcal{Z} \to \mathcal{P}(\chi)$ and $\tilde{f}|_{\mathcal{U}\times\mathcal{Z}} \cap \mathcal{P}(\mathcal{U}) = f$;

(ii) $\tilde{h} : \chi \times \mathcal{Z} \to \mathcal{Z}$ and $\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$.

The definition of interval compatible pair changes to the following definition.

Definition 5.1.3. Consider the N-extension $\tilde{\Sigma} = \mathcal{S}(\chi, \mathcal{Z}, \tilde{f}, \tilde{h})$ of the nondeterministic transition system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ on $(\chi, \leq)$. The pair $(\tilde{\Sigma}, (\chi, \leq))$ is said to be *N-interval compatible* if

(i) each $\tilde{\Sigma}$-transition class, $\mathcal{T}_i(\tilde{\Sigma}) \in \mathcal{T}(\tilde{\Sigma})$, is an interval sublattice of $(\chi, \leq)$:

$$\mathcal{T}_i(\tilde{\Sigma}) = [\bigwedge \mathcal{T}_i(\tilde{\Sigma}), \bigvee \mathcal{T}_i(\tilde{\Sigma})];$$

(ii) $\tilde{f} : ([w_1, w_2], z) \longrightarrow [\bigwedge \tilde{f}(w_1, z), \bigvee \tilde{f}(w_2, z)]$ is order preserving for any $[w_1, w_2] \subseteq \mathcal{T}_i(\tilde{\Sigma})$, and any $z \in \mathcal{Z}$ and for any $i \in \{1, ..., M\}$;

(iii) $\tilde{f} : ([w_1, w_2] \cap \mathcal{U}, z) \longrightarrow [\bigwedge \tilde{f}(w_1, z), \bigvee \tilde{f}(w_2, z))] \cap \mathcal{U}$ is onto for any $[w_1, w_2] \subseteq \mathcal{T}_i(\tilde{\Sigma})$ for any $z \in \mathcal{Z}$ and for any $i \in \{1, ..., M\}$.

Note that for a set-valued function f, we have that $f : A \to B$ is onto if for any element $b \in B$ there is an element $a \in A$ such that $b \in f(a)$. In the following section, the estimator is constructed.

5.2 Estimator Construction and Existence

In this section, we show how to extend the estimator construction and existence to nondeterministic systems. The arguments carried out are similar to the deterministic setting. The main difference lies in the fact that now we deal with set valued maps as opposed to single valued maps. This will be taken directly into account in the estimator construction. Also, this feature will no longer allow us to prove the monotonicity property of the estimation error that we had in the deterministic setting (see (ii) of Problem 3.2.1). In particular, Theorem 3.3.1 transforms to the following.

Theorem 5.2.1. *Assume that the nondeterministic transition system* $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ *is weakly observable. If there is a lattice* $(\chi, \leq)$*, such that the pair* $(\tilde{\Sigma}, (\chi, \leq))$ *is N-interval compatible, then the deterministic transition system with input* $\hat{\Sigma} = (\chi \times \chi, \mathcal{Z} \times \mathcal{Z}, \chi \times \chi, (f_1, f_2), id)$ *with*

$$\begin{aligned} f_1(L(k), y(k), y(k+1)) &= \bigwedge \tilde{f}(L(k) \vee \bigwedge O_y(k), y(k)) \\ f_2(U(k), y(k), y(k+1)) &= \bigvee \tilde{f}(U(k) \wedge \bigwedge O_y(k), y(k)) \end{aligned} \tag{5.2}$$

solves (i) and (iii) of Problem 3.2.1.

Proof. The proof is similar to the one of Theorem 3.3.1, except that now the function $\tilde{f}$ is nondeterministic, and thus one has to carry out the arguments using $\vee\tilde{f}$ and $\wedge\tilde{f}$ as opposed to $\tilde{f}$ itself. This is sketched in what follows. For simplifying notation, we omit the dependence of $\tilde{f}$ on y.

Item (i) can be proved by induction on k. By the initialization of the estimator $L(0) \leq \alpha(0) \leq U(0)$ (base case). Assume that $L(k) \leq \alpha(k) \leq U(k)$, and show this holds at step $k+1$. It suffices to notice that $\wedge O_y(k) \vee L(k) \leq \alpha(k) \leq U(k) \wedge \vee O_y(k)$, because $\alpha(k) \in O_y(k)$ by definition. By the order preserving property of $\tilde{f}$, we have

$$\bigwedge \tilde{f}(\bigwedge O_y(k) \vee L(k)) \leq \bigwedge \tilde{f}(\alpha(k)) \leq \alpha(k+1)$$

and

$$\alpha(k+1) \leq \bigvee \tilde{f}(\alpha(k)) \leq \bigvee \tilde{f}(U(k) \wedge \bigvee O_y(k)).$$

Item (iii) of Problem 3.2.1 is proved by contradiction. Assume $\beta_1'', \beta_2'' \in [L(k+1), U(k+1)] \cap \mathcal{U}$. By equations (5.2) and by property (iii) of Definition 5.1.3, there are $\beta_1', \beta_2' \in [\wedge O_y(k) \vee L(k), U(k) \wedge \vee O_y(k)] \cap \mathcal{U}$ such that $\beta_1'' \in f(\beta_1')$ and $\beta_2'' \in f(\beta_2')$, and $\beta_1', \beta_2' \in O_y(k)$. In an analogous way, there are $\beta_1, \beta_2 \in [\wedge O_y(k-1) \vee L(k-1), U(k-1) \wedge \vee O_y(k-1)] \cap \mathcal{U}$ such that $\beta_1' \in f(\beta_1)$ and $\beta_2' \in f(\beta_2)$, and $\beta_1, \beta_2 \in O_y(k-1)$. This implies that one can construct two executions of Σ, $\sigma_1 = \{\beta_1(k), z(k)\}_{k \in \mathbb{N}}$ and $\sigma_2 = \{\beta_2(k), z(k)\}_{k \in \mathbb{N}}$ that share the same output. This contradicts the weak observability of Σ. □

In Theorem 5.2.1, we assume that the system is weakly observable as opposed to observable as assumed in Theorem 3.3.1, and the functions f_1 and f_2 are modified by taking that $f(\cdot)$ is a set into account. Also, (ii) of Problem 3.2.1 cannot be guaranteed because $\tilde{f}$ maps an element to a set.

The following theorem shows that, just as in the case of deterministic systems, it is always possible to find a lattice $(\chi, \leq)$ and a system extension $\tilde{\Sigma}$ such that the pair $((\chi, \leq), \Sigma)$ is N-interval compatible.

Theorem 5.2.2. *Consider the nondeterministic system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$. There exists a lattice $(\chi, \leq)$, with $\mathcal{U} \subset \chi$, and extensions $\tilde{f} : \chi \times \mathcal{Z} \to \mathcal{P}(\chi)$, $\tilde{h} : \chi \times \mathcal{Z} \to \mathcal{Z}$, with $f = \tilde{f}|_{\mathcal{U}\times\mathcal{Z}} \cap \mathcal{P}(\mathcal{U})$ and $\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$, such that the pair $((\chi, \leq), \tilde{\Sigma})$ is N-interval compatible.*

Proof. The proof proceeds by construction. (0) A lattice $(\chi, \leq)$ with $\mathcal{U} \subset \chi$ is constructed; (1) the map $h : \mathcal{U} \times \mathcal{Z} \to \mathcal{Z}$ is extended to $(\chi, \leq)$ such that (i) is verified; (2) the map $f : \mathcal{U} \times \mathcal{Z} \to \mathcal{P}(\mathcal{U})$ is extended to $(\chi, \leq)$ such that $\tilde{f}|_{\mathcal{U}\times\mathcal{Z}} \cap \mathcal{P}(\mathcal{U}) = f$, and such that (ii)-(iii) of Definition 5.1.3 are verified. Since the constructions (0) and (1) are identical to the deterministic case, the proof concentrates on proving (2).

(2) In order to prove (ii) of Definition 5.1.3, $\tilde{f}$ is defined. For simplifying the notation, we omit the dependence on z. We define $\tilde{f}$ for every element $w \in \chi$. By the construction in (0), any $w \in \chi$ has the form $w = \alpha_1 \vee ... \vee \alpha_p$ for some $\alpha_i \in \mathcal{U}$. Thus, for every p-tuple $(\alpha_1, ..., \alpha_p)$, define

$$\begin{aligned} \tilde{f}(\alpha_1 \vee ... \vee \alpha_p) &:= \tilde{f}(\alpha_1) \vee ... \vee \tilde{f}(\alpha_p), \\ \tilde{f}(\alpha_1) \vee ... \vee \tilde{f}(\alpha_p) &:= \mathcal{P}((f(\alpha_1) \cup ... \cup f(\alpha_p)). \end{aligned} \tag{5.3}$$

This is shown in Figure 5.1 for $p = 2$. Therefore, for any $w \in \chi$, it follows that $\tilde{f}(w) = [\perp, \vee\tilde{f}(w)]$. Also define $\tilde{f}(\perp) = \perp$. It follows by construction that $f(\alpha) = \tilde{f}(\alpha) \cap \mathcal{U}$ for any $\alpha \in \mathcal{U}$. To show that $\tilde{f}$ is order preserving, we check Definition 2.1.3. Since for any $w \leq x$ we have $\wedge\tilde{f}(w) = \wedge\tilde{f}(x) = \perp$, we need to check that $\vee\tilde{f}(w) \leq \vee\tilde{f}(x)$. In fact if $w \leq x$, then $w = \alpha_1 \vee ... \vee \alpha_m$ and $x = \alpha_1 \vee ... \vee \alpha_m \vee \alpha_{m+1} \vee ... \vee \alpha_n$ for some $\alpha_i \in \mathcal{U}$ by part (0). By definition $\tilde{f}(w) = \mathcal{P}((f(\alpha_1) \cup ... \cup f(\alpha_m))$ and $\tilde{f}(x) = \mathcal{P}((f(\alpha_1) \cup ... \cup f(\alpha_m) ... \cup f(\alpha_n))$. Therefore $\tilde{f}(w) =$

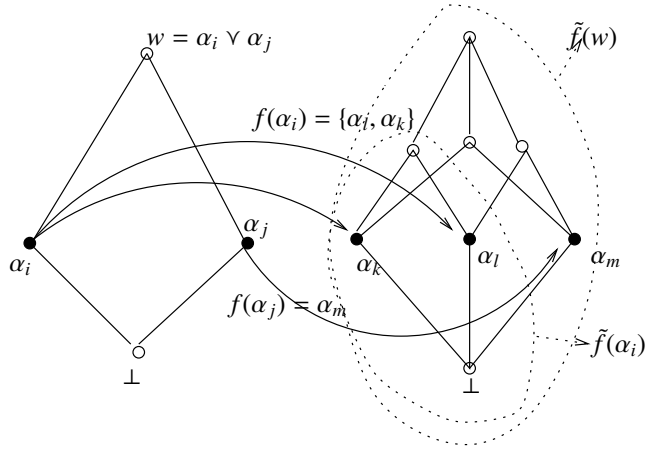


Figure 5.1: Extension $\tilde{f}$ on lattice χ.

$\cup_{j=1:m} f(\alpha_j) \subset \cup_{j=1:n} f(\alpha_j) = \bigvee \tilde{f}(x)$. (iii) To prove that $\tilde{f} : [\bot, U] \cap \mathcal{U} \longrightarrow [\bot, \vee\tilde{f}(U)] \cap \mathcal{U}$ is onto, we need to show that for any $\beta \in [\bot, \bigvee \tilde{f}(U)] \cap \mathcal{U}$ there is $\alpha \in [\bot, U] \cap \mathcal{U}$ such that $\beta \in \tilde{f}(\alpha)$. By construction (part (0)) we have that $U = \alpha_1 \vee ... \vee \alpha_n$, for some $\alpha_1, ..., \alpha_n$. Therefore $[\bot, \bigvee \tilde{f}(U)] \cap \mathcal{U} = \cup_{i=1:n} f(\alpha_i)$, which implies that $\beta \in f(\alpha_i)$ for some $i \in \{1, ..., n\}$. Since $U = \alpha_1 \vee ... \vee \alpha_n$ we have that $\alpha_i \in [\bot, U] \cap \mathcal{U}$ for all i. □

Note that an equivalent of Proposition 4.2.2 holds if Σ is nondeterministic and weakly observable. For completeness we reformulate such a proposition.

Proposition 5.2.3. *If the nondeterministic transition system $\Sigma = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f, h)$ is weakly observable and its Σ-weakly equivalent generalization $\Sigma_\geq = \mathcal{S}(\mathcal{U}_\geq, \mathcal{Z}, f_\geq, h)$ is such that the pair $(\tilde{\Sigma}_\geq, (\chi, \leq))$ is N-interval compatible for a given $(\chi, \leq)$, then Theorem 5.2.1 can be applied to $\Sigma_\geq$ with $\alpha(k) = \sigma_\Sigma(k)(\alpha)$ and $z(k) = \sigma_\Sigma(k)(z)$.*

In the following example, we show how to apply this proposition to a nondeterministic version of the RoboFlag system in order to construct an estimator.

5.3 Nondeterministic Example

In this section, we propose a non-deterministic version of the RoboFlag system, and we show how to construct an estimator. The system is analogous to the one analyzed in Section 3.4 except for the way the assignment is updated. In fact, we assume that at each step only

one pair of robots among the pairs with conflicting assignments swap the assignment, the pair that switches being randomly chosen. The blue robot dynamics are described by the rules

$$z_i(k+1) = z_i(k) + \delta \quad \text{if} \quad z_i(k) < x_{\alpha_i(k)} \tag{5.4}$$

$$z_i(k+1) = z_i(k) - \delta \quad \text{if} \quad z_i(k) > x_{\alpha_i(k)} \tag{5.5}$$

$$(\alpha_i(k+1), \alpha_{i+1}(k+1)) = (\alpha_{i+1}(k), \alpha_i(k)) \quad \text{if} \quad switch_{(i,i+1)}(k), \tag{5.6}$$

for $i \in \{1, ..., N\}$, where $switch_{(i,j)}(k)$ is such that

$$switch_{(i,i+1)}(k) \Rightarrow x_{\alpha_i(k)} \geq x_{\alpha_{i+1}(k)} \tag{5.7}$$

$$switch_{(i,i+1)}(k) \wedge switch_{(j,j+1)}(k) = false, \ i \neq j \tag{5.8}$$

$$\begin{aligned} &\left((x_{\alpha_1(k)} \geq x_{\alpha_2(k)}) \vee .... \vee (x_{\alpha_{N-1}(k)} \geq x_{\alpha_N(k)})\right) \Rightarrow \\ &\quad (switch_{(1,2)}(k) \vee ... \vee switch_{(N-1,N)}(k) = true. \end{aligned} \tag{5.9}$$

Rules (5.6) establish that two close robots will exchange their assignments if *switch* is true. In particular, (5.7) implies that *switch* can be true only for two robots with conflicting assignments, (5.9) establishes that one pair of close robots will exchange assignments provided there is at least one pair of robots with conflicting assignments, and (5.8) implies that only one pair of robots will exchange assignments. Therefore (5.7), (5.8), and (5.9), along with (5.6), guarantee that, if there are some close robots with conflicting assignments, there is one and only one pair of robots among them that will switch the assignments. This renders the assignment protocol in commands (5.6) nondeterministic, as at each step we do not know which pair of robots switches assignments. It is possible to show that the assignment protocol converges to the equilibrium value $(1, ..., N)$. For this, we defer the reader to [35]. For the blue robots, we assume that initially $z_i \in [z_{min}, z_{max}]$, $z_i < z_{i+1}$, and that $x_i < z_i < x_{i+1}$ for all time. With this assumption, one can check that system Σ is weakly observable. The proof is similar to the one given in Proposition 3.4.1. We define $x = (x_1, ..., x_N)$, $z = (z_1, ..., z_N)$, $\alpha = (\alpha_1, ..., \alpha_N)$.

The rules reported in (5.6) determine the function $f : \mathcal{U} \times \mathbb{R}^N \to \mathcal{P}(\mathcal{U})$ that updates the

discrete variables α, while the rules in (5.5) and (5.4) determine the function $h : \mathcal{U} \times \mathbb{R}^N \to \mathbb{R}^N$. Therefore the blue robot system is defined by $\Sigma = \mathcal{S}(\text{perm}(N), \mathbb{R}^N, f, h)$.

For the purpose of constructing the estimator, we consider the order $(\chi, \leq)$ described in Section 3.4.2. One can verify that there is no extension of Σ on χ that is N-interval compatible with $(\chi, \leq)$. As a consequence, we apply Proposition 5.2.3. We look for a Σ-weakly equivalent generalization of the NTS Σ that admits an extension $\tilde{\Sigma}_{\geq}$ on χ that is N-interval compatible with $(\chi, \leq)$. We define the system $\Sigma_{\geq} = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f_{\geq}, h)$ by defining $f_{\geq}$ in the following way. Let $v(k) = z(k+1) - z(k)$ denote the velocity, then at step k we have for $\beta \in \mathcal{U}$

$$f_{\geq}(\beta, z) := f(\beta, z) \text{ if } v(k) \neq v(k-1) \tag{5.10}$$

$$f_{\geq}(\beta, z) := [\bigwedge O_y(k), \bigvee O_y(k)] \cap \mathcal{U} \text{ otherwise.} \tag{5.11}$$

It is easy to verify (i)-(ii) of Definition 4.2.1, so that $\Sigma_{\geq} = \mathcal{S}(\mathcal{U}, \mathcal{Z}, f_{\geq}, h)$ is a Σ-weakly equivalent generalization of $\Sigma = \mathcal{S}(\text{perm}(N), \mathbb{R}^N, f, h)$. Property (i) is trivially verified. To verify (ii) it is enough to notice that the switch before the stabilization time k_σ of the sequence $\{\sigma_\Sigma(k)(\alpha)\}_{k\in\mathbb{N}}$ is observable. Let $\sigma_{\Sigma_{\geq}}$ denote an execution of $\Sigma_{\geq}$ and $\{\sigma_{\Sigma_{\geq}}(k)(\alpha)\}_{k\in\mathbb{N}}$ the corresponding α sequence; we have that $f_{\geq}(\sigma_{\Sigma_{\geq}}(k_\sigma - 1)(\alpha), z(k_\sigma - 1)) = f(\sigma_{\Sigma_{\geq}}(k_\sigma - 1)(\alpha), z(k_\sigma - 1)) = (1, ..., N)$. This in turn implies that $\sigma_{\Sigma_{\geq}}(k_\sigma - 1)(\alpha) = \sigma_\Sigma(k_\sigma - 1)(\alpha)$ for some execution σ_Σ of Σ.

To find an extension $\tilde{\Sigma}_{\geq}$ that is N-interval compatible with $(\chi, \leq)$, consider the following extension of $f_{\geq}$ on χ at step k for any $q \in \chi$

$$\tilde{f}_{\geq}(q, z) = w, \ (w_i, w_{i+1}) := (q_{i+1}, q_i), \text{ if } v_i(k) \neq v_i(k-1) \tag{5.12}$$

$$\tilde{f}_{\geq}(q, z) := [\bigwedge O_y(k), \bigvee O_y(k)] \text{ otherwise.} \tag{5.13}$$

Expression (5.12) defines an order isomorphism between $[L, U]$ and $[\tilde{f}_{\geq}(L, z), \tilde{f}_{\geq}(U, z)]$ for any $L, U \in \chi$. From expression (5.13), we deduce that $\tilde{f}_{\geq}$ is trivially order preserving according to the Definition 2.1.3. Moreover $\tilde{f}_{\geq} : ([L, U] \cap \mathcal{U}, z) \to [\bigwedge \tilde{f}_{\geq}(L, z), \bigvee \tilde{f}_{\geq}(U, z)] \cap \mathcal{U}$ is clearly onto by construction for any $[L, U] \subseteq O_y(k)$, and $\tilde{f}_{\geq}|_{\mathcal{U}} \cap \mathcal{P}(\mathcal{U})$ coincides with

$f_{\geq}$ by construction as well. As a consequence the system $\tilde{\Sigma}_{\geq} = \mathcal{S}(\mathcal{P}(\chi), \mathcal{Z}, \tilde{f}_{\geq}, \tilde{h})$ with $\tilde{h}$ as defined in Section 3.4.2 is N-interval compatible with $(\chi, \leq)$.

We then apply Proposition 5.2.3 for constructing the estimator. Such an estimator can be written as a set of rules as already done for the example in Section 3.4.2. In Figure 5.2 we report

$$W(k) = \frac{1}{N} \sum_{i=1}^{N} |m_i(k)|,$$

which converges to 1 when the value of α has been locked, and

$$E(k) = \frac{1}{N} \sum_{i=1}^{N} |\alpha_i(k) - i|,$$

which gives an idea of the speed of convergence of the assignment to the equilibrium value $(1, ..., N)$. $|[L(k), U(k)] \cap \mathcal{U}|$ converges to 1, but $|[L(k), U(k)]|$ is not a monotonic function of k as it was in the deterministic case. This is due to the nondeterministic nature of the transition functions, as one element can be mapped to many. The choice of $f_{\geq}$ has a considerable impact on the convergence speed of the estimator. The map $f_{\geq}$ we chose is rough and does not take other information that we have on the system into account. For example it does not model the fact that even if there is an unobservable switch, a subset of the robots, depending on their assignment estimates, undergoes particular switches. The more information we can model with $f_{\geq}$ the faster the convergence rate.

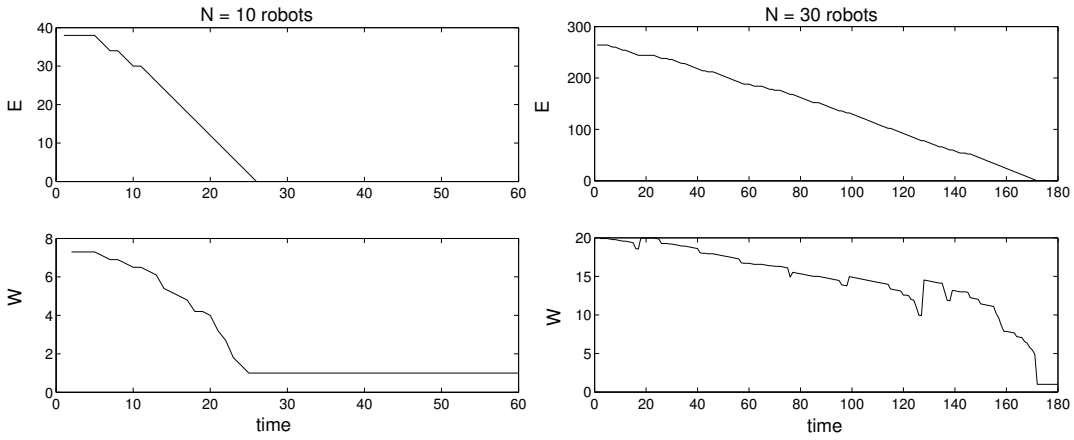


Figure 5.2: Example with N=10(left) and N=30(right): upper plot shows the stabilization function of the α assignment (E), while lower plot shows the function W for the estimator.

Chapter 6

Cascade Discrete-Continuous State Estimators on a Lattice

In this chapter, a cascade discrete-continuous state estimator on a partial order is proposed and its existence investigated. The continuous state estimation error is bounded by a monotonically non-increasing function of the discrete state estimation error, with both the estimation errors converging to zero. We show that the lattice approach to estimation is general as the proposed estimator can be constructed for any observable and independent discrete state observable system. The main advantage of using the lattice approach for estimation becomes clear when the system has monotone properties that can be exploited in the estimator design. In such a case, the computational complexity of the estimator can be drastically reduced and tractability can be achieved. Some examples are proposed to illustrate these ideas. This chapter is structured as follows. In Section 6.1, we introduce the model of the system, which is equal to the one of the previous chapters except that now the continuous variables are not measured. In Section 6.2, the problem is formulated, and in Section 6.3 a solution is proposed. In Section 6.4, we show that under suitable observability assumptions, the proposed estimator can always been constructed. In Section 6.5, we show a particular class of systems, monotone systems, for which the order relations can be efficiently computed at least in the continuous variable space. In Section 6.6, three examples with decreasing complexity are shown, and the complexities of the respective estimators are computed. The results of this chapter appeared in [25].

6.1 The Model

The model that we consider in this chapter is the same as the one in Section 3.2, except that now the continuous variables are not measured anymore. Then, for a system $\Sigma = (S, \mathcal{Y}, F, g)$, suppose that

(i) $S = \mathcal{U} \times \mathcal{Z}$ with $\mathcal{U}$ a finite set, and $\mathcal{Z}$ an infinite possibly dense set, and $\mathcal{Y}$ is a finite or infinite set;

(ii) $F = (f, h)$, where $f : \mathcal{U} \times \mathcal{Y} \to \mathcal{U}$ and $h : \mathcal{U} \times \mathcal{Z} \to \mathcal{Z}$;

(iii) $g : \mathcal{U} \times \mathcal{Z} \to \mathcal{Y}$ is the output map.

These systems have the form

$$\begin{aligned} \alpha(k+1) &= f(\alpha(k), y(k)) && (6.1)\\ z(k+1) &= h(\alpha(k), z(k)) && (6.2)\\ y(k) &= g(\alpha(k), z(k)), \end{aligned}$$

and they are referred to as the tuple $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$. The function f that updates the discrete variable α can be represented by a set of logic statements, or, in the case $\mathcal{Y}$ is finite, by a look-up table or recursive formula as is the case of finite state machines ([32]). For each value of α, the equation (6.2) is a difference equation. The set $\mathcal{Y}$ can be such that the output has both a continuous and a discrete component. We leave $\mathcal{Y}$ unspecified for the moment. In the examples at the end of the chapter, we will show several forms that this set can take.

Before stating the problem in more detail, we recall the notions of transition sets of Σ given in Section 4.1, by redefining them for the present case in which the variables z are not measured.

Definition 6.1.1. (Σ-transition sets) The non empty sets $T_{(y_1,y_2)}(\Sigma) = \{\alpha \in \mathcal{U} \mid y_1 = g(\alpha, z) \text{ and } y_2 = g(f(\alpha, y_1), h(\alpha, z))\}$, are the Σ-*transition sets*.

In general these sets depend on z. For the sake of simplicity, we are interested in the case in which these sets do not depend on z. Thus, we give the following definition.

Definition 6.1.2. (Independent discrete state observability) The system $\Sigma = (\mathcal{U}\times\mathcal{Z}, \mathcal{Y}, (f, h), g)$ is said to be *independent discrete state observable* if for any execution with output sequence $\{y(k)\}_{k\in\mathbb{N}}$, the following are verified

(i) the Σ-transition sets $T_{(y(k),y(k+1))}(\Sigma)$ do not depend on z;

(ii) for any two executions $\sigma_1, \sigma_2 \in \mathcal{E}(\Sigma)$ with $g(\sigma_1(k)) = g(\sigma_2(k)) = y(k)$ and with $\{\sigma_1(k)(\alpha)\}_{k\in\mathbb{N}} \neq \{\sigma_2(k)(\alpha)\}_{k\in\mathbb{N}}$, there is $k > 0$ such that $\sigma_1(k)(\alpha) \in T_{(y(k),y(k+1))}(\Sigma)$ and $\sigma_2(k)(\alpha) \notin T_{(y(k),y(k+1))}(\Sigma)$.

Item (i) is trivially verified if $g(\alpha, z) = (g_\alpha(\alpha), g_z(\alpha, z))$, where $g_\alpha : \mathcal{U} \to \{Y_1, Y_2, ..., Y_m\}$ partitions the set $\mathcal{U}$ in equivalence classes. We allow two steps in order to have an equivalence class that is independent of $z(k)$, as this is often the case when α acts in the z dynamics. This assumption is made for the sake of simplicity. It can be relaxed to allow a finite number of N steps for obtaining a set $T_{(y(k),y(k+1),...,y(k+N))}(\Sigma)$ that does not depend on z with minor modifications to the estimator structure. From this definition, it follows that an independent discrete state observable system admits a discrete state estimator that does not involve the continuous state estimate. This property allows us to construct a cascade discrete-continuous state estimator, that is, an estimator in which the discrete state estimate is done as in Chapter 3, and the continuous state estimate is a function of the discrete state estimate. This is formally explained in the following section.

6.2 Problem Statement

Consider the deterministic transition system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$, with the output sequence $\{y(k)\}_{k\in\mathbb{N}}$. The problem of determining and tracking the value of the current state $(\alpha(k), z(k))$ of the system is formally stated in the following problem.

Problem 6.2.1. (Cascade discrete-continuous state estimator) Given the deterministic transition system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$, find a deterministic transition system with input $\hat{\Sigma} = (\chi \times \chi \times \mathcal{L} \times \mathcal{L}, \mathcal{Y} \times \mathcal{Y}, \chi \times \chi \times \mathcal{Z}_E \times \mathcal{Z}_E, (f_1, f_2, f_3, f_4), (f_5, f_5, f_6, f_7))$, with $f_1 : \chi\times\mathcal{Y}\times\mathcal{Y} \to \chi$, $f_2 : \chi\times\mathcal{Y}\times\mathcal{Y} \to \chi$, $f_3 : \mathcal{L}\times\chi\times\mathcal{Y}\times\mathcal{Y} \to \mathcal{L}$, $f_4 : \mathcal{L}\times\chi\times\mathcal{Y}\times\mathcal{Y} \to \mathcal{L}$,

$f_5 : \chi \to \chi$ and $f_5 = \mathrm{id}$, $f_6 : \mathcal{L} \to \mathcal{Z}_E$, with $\mathcal{U} \subseteq \chi$, $(\chi, \leq)$ a lattice, $\mathcal{Z} \subseteq \mathcal{Z}_E$ with $(\mathcal{Z}_E, \leq)$ a lattice, $\chi \times \mathcal{Z}_E \subseteq \mathcal{L}$, $(\mathcal{L}, \leq)$ a lattice, such that the update laws

$$\begin{aligned} L(k+1) &= f_1(L(k), y(k), y(k+1)) \\ U(k+1) &= f_2(U(k), y(k), y(k+1)) \\ q_L(k+1) &= f_3(q_L(k), L(k), y(k), y(k+1)) \\ q_U(k+1) &= f_4(q_U(k), U(k), y(k), y(k+1)) \\ z_L(k) &= f_6(q_L(k)) \\ z_U(k) &= f_7(q_U(k)) \end{aligned} \tag{6.3}$$

with $L(k), U(k) \in \chi$, $L(0) := \bigwedge \chi$, $U(0) := \bigvee \chi$, $q_L(k), q_U(k) \in \mathcal{L}$, $q_L(0) = \bigwedge \mathcal{L}$, $q_U(0) = \bigvee \mathcal{L}$, and $z_L(k), z_U(k) \in \mathcal{Z}_E$, have the following properties

(i) $L(k) \leq \alpha(k) \leq U(k)$ (correctness);

(ii) $|[L(k+1), U(k+1)]| \leq |[L(k), U(k)]|$ (non-increasing error);

(iii) there exists $k_0 > 0$ such that $[L(k), U(k)] \cap \mathcal{U} = \alpha(k)$ for any $k \geq k_0$ (convergence);

(i') $z_L(k) \leq z(k) \leq z_U(k)$;

(ii') there is a nonnegative function $V : \mathbb{N} \to \mathbb{R}$ such that $d(z_L(k), z_U(k)) \leq V(k)$, with $V(k+1) \leq V(k)$;

(iii') there exists $k_0' > 0$ such that $d(z_{L'}(k), z_{U'}(k)) = 0$ for any $k \geq k_0'$, where

$$\begin{aligned} L' &= \bigwedge([L, U] \cap \mathcal{U}) \\ U' &= \bigvee([L, U] \cap \mathcal{U}) \\ q_{L'}(k+1) &= f_3(q_{L'}(k), L'(k), y(k), y(k+1)) \\ q_{U'}(k+1) &= f_4(q_{U'}(k), U'(k), y(k), y(k+1)) \\ z_{L'}(k) &= f_6(q_{L'}(k)) \\ z_{U'}(k) &= f_7(q_{U'}(k)), \end{aligned}$$

with $q_{L'}(0) = q_L(0)$ and $q_{U'}(0) = q_U(0)$, for some distance function "d."

□

The update laws (6.3) are in cascade form as the variables L and U are updated on the basis of their previous values and on the basis of the output, while the variables q_L and q_U are updated on the basis of their previous values, on the basis of the output, and on the basis of the values of L and U, respectively. Note that the lower and the upper bound estimates of $z(k)$ are outputs of the laws that update $q_L(k)$ and $q_U(k)$, which lie in the space $\mathcal{L}$. Properties (iii) and (iii') ask that the lower and upper bounds converge to $\alpha(k)$ and $z(k)$. Property (ii') gives a monotonic bound on the continuous variable estimation error.

Note that the distance function "d" has been left unspecified for the moment, as its form depends on the particular partial order chosen $(\mathcal{Z}_E, \leq)$. In the case in which $\mathcal{Z}_E = \mathcal{Z}$ and $\mathcal{Z} = \mathbb{R}^n$ with the order is established component-wise, the distance can be the classical euclidean distance. In the following section, a solution to Problem 6.2.1 is proposed.

6.3 Estimator Construction

Given the deterministic transition system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$, a set of sufficient conditions that allow a solution to Problem 6.2.1 is provided. With this respect, some definitions involving the extension of the system Σ to a lattice are useful.

Definition 6.3.1. (Extended system) Consider the system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$. Let $(\chi, \leq)$, $(\mathcal{Z}_E, \leq)$, and $(\mathcal{L}, \leq)$ be lattices with $\mathcal{U} \subseteq \chi$, $\mathcal{Z} \subseteq \mathcal{Z}_E$, and $\chi \times \mathcal{Z}_E \subseteq \mathcal{L}$. An extension of Σ on the lattice $(\mathcal{L}, \leq)$ is given by $\tilde{\Sigma} = (\mathcal{L}, \mathcal{Y}, \tilde{F}, \tilde{g})$ such that

(i) $\tilde{F} : \mathcal{L} \times \mathcal{Y} \to \mathcal{L}$ and $\tilde{F}|_{\mathcal{U}\times\mathcal{Z}\times\mathcal{Y}} = (f, h)$, and $\mathcal{L} - (\mathcal{U} \times \mathcal{Z})$ is invariant under $\tilde{F}$;

(ii) $\tilde{F}|_{\chi\times\mathcal{Z}_E\times\mathcal{Y}} = (\tilde{f}, \tilde{h})$, where $\tilde{f} : \chi \times \mathcal{Y} \to \chi$, $\tilde{f}|_{\mathcal{U}\times\mathcal{Y}} = f$, $\tilde{h} : \chi \times \mathcal{Z}_E \to \mathcal{Z}_E$, and $\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$;

(iii) $\tilde{g} : \mathcal{L} \to \mathcal{Y}$ and $\tilde{g}|_{\mathcal{U}\times\mathcal{Z}} = g$;

(iv) the partial order $(\mathcal{L}, \leq)$ is closed with respect to $\chi \times \mathcal{Z}_E$.

Item (iv) of the above definition establishes (according to Definition 2.1.6) that the chosen lattices are such that any element in $\mathcal{L}$ that is not in $\chi \times \mathcal{Z}_E$ can be approximated by two elements in $\chi \times \mathcal{Z}_E$, a lower element $a_L(q)$ and an upper element $a_U(q)$. These are the lower and upper approximations of q, respectively. Note that if $q \in \chi \times \mathcal{Z}_E$, then $a_L(q) = a_U(q) = q$.

In the following two definitions, we redefine the notions of $\tilde{\Sigma}$-transition sets and of $\tilde{\Sigma}$-transition classes.

Definition 6.3.2. ($\tilde{\Sigma}$-transition sets) Let $\tilde{\Sigma} = (\mathcal{L}, \mathcal{Y}, \tilde{F}, \tilde{g})$ be the extension of Σ on the lattice $(\mathcal{L}, \leq)$. For any $y_1, y_2 \in \mathcal{Y}$, the non-empty sets $T_{(y_1,y_2)}(\tilde{\Sigma}) = \{w \in \chi \mid y_2 = \tilde{g}(\tilde{f}(w, y_1), \tilde{h}(w, z))$ and $y_1 = \tilde{g}(w, z)\}$ for $z \in \mathcal{Z}$ are named the $\tilde{\Sigma}$-*transition sets*.

By the independent discrete state observability property, the Σ-transition sets do not depend on z. One can always obtain this same property for the $\tilde{\Sigma}$-transition sets by a proper definition of the system extension on χ. In the sequel, we assume that the system extension $\tilde{\Sigma}$ has been chosen so that also the $\tilde{\Sigma}$-transition sets do not depend on z. As done in Section 3.2, we can also define the $\tilde{\Sigma}$ transition classes. We recall this definition for completeness.

Definition 6.3.3. ($\tilde{\Sigma}$-Transition classes) The set $\mathcal{T}(\tilde{\Sigma}) = \{\mathcal{T}_1(\tilde{\Sigma}), ..., \mathcal{T}_M(\tilde{\Sigma})\}$, with $\mathcal{T}_i(\tilde{\Sigma})$ such that

(i) for any $\mathcal{T}_i(\tilde{\Sigma}) \in \mathcal{T}(\tilde{\Sigma})$ there are $y_1, y_2 \in \mathcal{Y}$ such that $\mathcal{T}_i(\tilde{\Sigma}) = T_{(y_1,y_2)}(\tilde{\Sigma})$;

(ii) for any $T_{(y_1,y_2)}(\tilde{\Sigma})$ there is $j \in \{1, ..., M\}$ such that $T_{(y_1,y_2)}(\tilde{\Sigma}) = \mathcal{T}_j(\tilde{\Sigma})$;

is the *set of* $\tilde{\Sigma}$-*transition classes*.

The next definition links the discrete state dynamics of $\tilde{\Sigma}$ with the partial order $(\chi, \leq)$ as done already in Chapter 3.

Definition 6.3.4. (Interval compatibility) The pair $(\tilde{\Sigma}, (\chi, \leq))$ is said to be *interval compatible* if the following are verified

(i) each $\tilde{\Sigma}$-transition class, $\mathcal{T}_i(\tilde{\Sigma}) \in \mathcal{T}(\tilde{\Sigma})$, is an interval sublattice of $(\chi, \leq)$:

$$\mathcal{T}_i(\tilde{\Sigma}) = [\bigwedge \mathcal{T}_i(\tilde{\Sigma}), \bigvee \mathcal{T}_i(\tilde{\Sigma})];$$

(ii) $\tilde{f} : (\mathcal{T}_i(\tilde{\Sigma}), y) \rightarrow [\tilde{f}(\bigwedge \mathcal{T}_i(\tilde{\Sigma}), y), \tilde{f}(\bigvee \mathcal{T}_i(\tilde{\Sigma}), y)]$ is an order isomorphism for any $i \in \{1, ..., M\}$ and for any $y \in \mathcal{Y}$.

This definition is the same as Definition 3.3.5, the only slight difference is in the way the transition classes have been defined due to the fact that now the z variables are not measured. Item (i) in the above definition implies that the set $T_{y(k),y(k+1)}(\tilde{\Sigma})$ of $w \in \chi$ compatible with the pair $(y(k), y(k+1))$ for any execution σ with output sequence $\{y(k)\}_{k\in\mathbb{N}}$ is a sublattice interval in χ. Also, the output set $O_y(k)$ is still given as in Chapter 3 by

$$O_y(k) = T_{y(k),y(k+1)}(\tilde{\Sigma}).$$

For the construction of a cascade discrete-continuous state estimator, the case in which the partial order $(\mathcal{L}, \leq)$ is induced by the partial order $(\chi, \leq)$ by means of the system dynamics is of interest. Thus, the notion of induced transition sets is introduced in the following definition.

Definition 6.3.5. (Induced transition sets) Consider the system $\tilde{\Sigma} = (\mathcal{L}, \mathcal{Y}, \tilde{F}, \tilde{g})$ and a transition set $T_{(y_1,y_2)}(\tilde{\Sigma})$ for some $y_1, y_2 \in \mathcal{Y}$. For any $w_1, w_2 \in T_{(y_1,y_2)}(\tilde{\Sigma})$ with $w_1 \leq w_2$, the non-empty sets

$$T^{(w_1,w_2)}_{(y_1,y_2)}(\tilde{\Sigma}) = \{q \in \mathcal{L} \mid \pi_1 \circ a_L(q) \geq w_1,\ \pi_1 \circ a_U(q) \leq w_2,\ y_2 = \tilde{g}(\tilde{F}(q, y_1)),\ \text{and } y_1 = \tilde{g}(q)\}$$

are named the induced transition sets.

Note that the number of such sets is not necessarily finite as $q \in \mathcal{L}$, $(\chi \times \mathcal{Z}_E) \subseteq \mathcal{L}$, and $\mathcal{Z}_E$ is not always finite. In analogy to how we have proceeded for the estimation of the discrete state, we consider the case in which the induced transition sets induced by an interval $[w_1, w_2] \subseteq \chi$ are themselves intervals in $\mathcal{L}$. In such a case, we say that the system $\tilde{\Sigma}$ and the partial order $(\mathcal{L}, \leq)$ are induced interval compatible. This concept is formally defined in the following definition.

Definition 6.3.6. (Induced interval compatibility) The pair $(\tilde{\Sigma}, (\mathcal{L}, \leq))$ is said to be *induced interval compatible* if for any $w_1, w_2 \in T_{y_1,y_2}(\tilde{\Sigma})$ for some $y_1, y_2 \in \mathcal{Y}$ with $w_1 \leq w_2$, we have that

(i) the induced transition sets are such that

$$\begin{aligned} \bigwedge T^{(w_1,w_2)}_{(y_1,y_2)} &= l_q(w_1) \\ \bigvee T^{(w_1,w_2)}_{(y_1,y_2)} &= u_q(w_2) \end{aligned}$$

with $l_q(w_1)$ and $u_q(w_2)$ such that $a_L(l_q(w_1)) = (w_1, l_z(w_1))$ and $a_U(u_q(w_2)) = (w_2, u_z(w_2))$, for $l_z(w_1), u_z(w_2) \in \mathcal{Z}_E$;

(ii) $\tilde{F} : ([l_q(w_1), u_q(w_2)], y_1) \to [\tilde{F}(l_q(w_1), y_1), \tilde{F}(u_q(w_2), y_1)]$ is order preserving, and $\tilde{F} : (\alpha \times [l_z(\alpha), u_z(\alpha)], y_1) \to [\tilde{F}(\alpha, l_z(\alpha), y_1), \tilde{F}(\alpha, u_z(\alpha), y_1)]$ is order isomorphic;

(iii) for any $[w_1, w_2] \subseteq T_{y_1,y_2}(\tilde{\Sigma})$, we have that

$$d\left(\pi_2 \circ a_L \circ \tilde{F}(l_q(w_1), y_1), \pi_2 \circ a_U \circ \tilde{F}(u_q(w_2), y_1)\right) \leq \gamma(|[w_1, w_2]|),$$

for some distance function "d," and $\gamma : \mathbb{N} \to \mathbb{R}$ is a monotonic function of its argument.

Item (i) of this definition means that a sublattice interval $[w_1, w_2] \subseteq \chi$ compatible with an output pair (y_1, y_2) induces a sublattice interval in $(\mathcal{L}, \leq)$ corresponding to the same output pair. Also, such output interval is approximated by the Cartesian product of the two sublattice intervals $[w_1, w_2] \subseteq \chi$ and $[l_z(w_1), u_z(w_2)] \subseteq \mathcal{Z}_E$, in which l_z depends only on w_1 and u_z depends only on w_2. Item (ii) establishes the usual order preserving properties of the extension, and item (iii) establishes that the size of the interval sublattice in $(\mathcal{Z}_E, \leq)$ induced by an interval $[w_1, w_2] \in \chi$ increases with the size of $[w_1, w_2]$. A solution to Problem 6.2.1 is proposed by the following theorem.

Theorem 6.3.1. *Given the system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$, assume that there are lattices $(\chi, \leq)$, $(\mathcal{Z}_E, \leq)$, and $(\mathcal{L}, \leq)$, with $\mathcal{U} \subseteq \chi$, $\mathcal{Z} \subseteq \mathcal{Z}_E$, and $\chi \times \mathcal{Z}_E \subseteq \mathcal{L}$ such that the pairs $(\tilde{\Sigma}, (\chi, \leq))$ and $(\tilde{\Sigma}, (\mathcal{L}, \leq))$ are interval compatible and induced interval compatible,*

respectively. Then a solution to Problem 6.2.1 is provided by

$$
\begin{aligned}
f_1(L(k), y(k), y(k+1)) &= \tilde{f}(\bigwedge O_y(k) \vee L(k), y(k)) \\
f_2(U(k), y(k), y(k+1)) &= \tilde{f}(\bigvee O_y(k) \wedge U(k), y(k)) \\
f_3(q_L(k), L(k), y(k), y(k+1)) &= \tilde{F}(q_L(k) \vee l_q(\bigwedge O_y(k) \vee L(k)), y(k)) \\
f_3(q_U(k), U(k), y(k), y(k+1)) &= \tilde{F}(q_U(k) \wedge u_q(\bigvee O_y(k) \wedge U(k)), y(k)) \\
f_6(q_L(k)) &= \pi_2 \circ a_L(q_L(k)) \\
f_7(q_U(k)) &= \pi_2 \circ a_U(q_U(k)).
\end{aligned}
$$

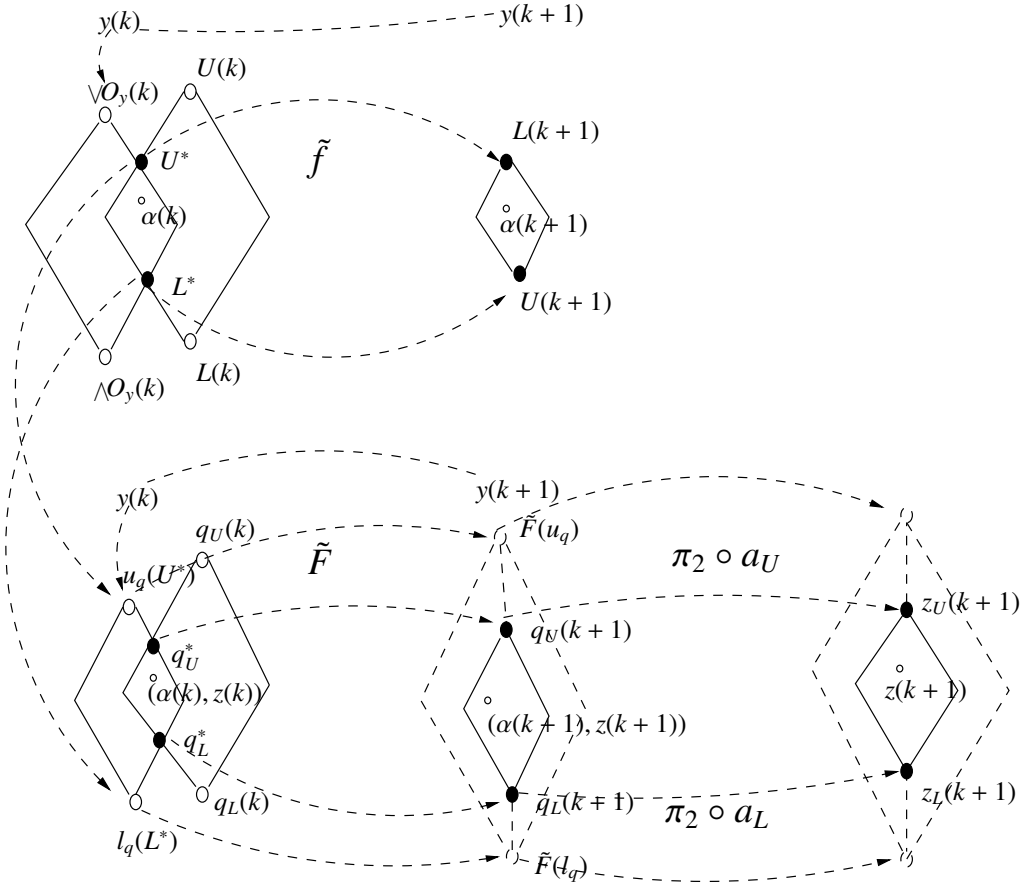


Figure 6.1: Hasse diagrams representing the updates of the estimator in Theorem 6.3.1. In the diagram, we have denoted $U^* = \bigvee O_y(k) \wedge U(k)$, $L^* = \bigwedge O_y(k) \vee L(k)$, $q_U^* = q_U(k) \wedge l_q(U^*)$, and $q_L^* = q_L(k) \vee l_q(L^*)$.

Proof. The idea of the proof is analogous to the one proposed in Theorem 3.3.1. Here,

a sketch is provided. For the proof of (i)-(ii)-(iii), the reader is deferred to the proof of Theorem 3.3.1. Define $U^* = \wedge O_y(k) \wedge U(k)$, $L^* = \vee O_y(k) \vee L(k)$, $q_U^* = q_U(k) \wedge l_q(U^*)$, and $q_L^* = q_L(k) \vee l_q(L^*)$. The dependence of u_q and l_q on their arguments is omitted, as well as the dependence of $\tilde{F}$ on y.

Proof of (i'). By using the induction argument on k and exploiting the order preserving property of $\tilde{F}$, one can show that $q_L(k) \leq (\alpha(k), z(k)) \leq q_U(k)$ (see Figure 6.1) for any k. By the the fact that $\pi_2 \circ a_L$ and $\pi_2 \circ a_U$ are order preserving functions, (i') follows (see Figure 6.1).

Proof of (ii'). Using the order preserving property of $\tilde{F}$, of $\pi_2 \circ a_L$, and of $\pi_2 \circ a_U$, one deduces that $z_L(k+1) \geq \pi_2 \circ a_L \circ \tilde{F}(l_q(L^*))$ and $z_U(k+1) \leq \pi_2 \circ a_U \circ \tilde{F}(u_q(U^*))$ (see Figure 6.1). By exploiting the property (iii) of the distance function in Definition 2.1.4 and the property (iv) given in Definition 6.3.6, one can infer that $d(z_L(k+1), z_U(k+1)) \leq \gamma(|[L^*, U^*]|)$. Since $\tilde{f}$ is order isomorphic, it follows that $|[L^*, U^*]| = |[\tilde{f}(L^*, y), \tilde{f}(U^*, y)]|$. Thus, (ii') of Problem 3.2.1 is satisfied with $V(k) = \gamma(|[L(k), U(k)]|)$.

Proof of (iii'). For $k > k_0$, $L'(k) = \alpha(k) = U'(k)$ as $[L(k), U(k)] \cap \mathcal{U} = \alpha(k)$. As a consequence, $q_{L'}(k+1) = \tilde{F}(q_{L'}(k) \vee l_q(\alpha(k)))$ and $q_{U'}(k+1) = \tilde{F}(q_{U'}(k) \vee u_q(\alpha(k)))$, where $l_q(\alpha) = (\alpha, l_z(\alpha))$ and $u_q(\alpha) = (\alpha, u_z(\alpha))$. One then uses the facts that $(\alpha, l_z(\alpha)) \leq q_{L'}(k) \vee l_q(\alpha(k))$, $q_{U'}(k) \vee u_q(\alpha(k)) \leq (\alpha, u_z(\alpha))$, the fact that $\tilde{F} : (\alpha \times [l_z(\alpha), u_z(\alpha)]) \to [\tilde{F}(\alpha, l_z(\alpha)), \tilde{F}(\alpha, u_z(\alpha))]$ is order isomorphic, and the fact that $\mathcal{L} - (\mathcal{U} \times \mathcal{Z})$ is invariant under $\tilde{F}$. Proceeding by contradiction, if for any k there are (α', z_1'), (α', z_2') in $[q_{L'}(k), q_{U'}(k)] \cap (\mathcal{U} \times \mathcal{Z})$ that are compatible with the output, there must be $(\alpha, z_1), (\alpha, z_2) \in [q_{L'}(k-1), q_{U'}(k-1)] \cap (\mathcal{U} \times \mathcal{Z})$ such that $(\alpha', z_1') = F(\alpha, z_1)$ and $(\alpha', z_2') = F(\alpha, z_2)$. Also, $(\alpha, z_1), (\alpha, z_2)$ are compatible with the output as well (see Figure 6.1). Since this is true for any k, one can construct two executions of Σ that are different and share the same output sequence. This contradicts observability of Σ. Then there must be $k > k_0$ such that $[q_{L'}(k), q_{U'}(k)] \cap (\mathcal{U} \times \mathcal{Z}) = (\alpha(k), z(k))$, and therefore $z_{L'}(k) = z_{U'}(k) = z(k)$. □

In the following section, conditions in order to verify the assumptions needed for the construction of the estimator given in Theorem 6.3.1 are given. In particular, observability and independent discrete state observability are sufficient conditions for the estimator

construction, and therefore the proposed estimation approach on a lattice is general.

6.4 Estimator Existence

The following theorem shows that if the system Σ is observable and independent discrete state observable, the lattices $(\mathcal{L}, \leq)$, $(\mathcal{Z}_E, \leq)$, and $(\chi, \leq)$ introduced in the previous section exist, such that the extended system is both interval compatible with $(\chi, \leq)$ and induced interval compatible with $(\mathcal{L}, \leq)$.

Theorem 6.4.1. *Assume that the system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$ is observable and independent discrete state observable. Then there exist lattices $(\chi, \leq)$, $(\mathcal{Z}_E, \leq)$, $(\mathcal{L}, \leq)$ with $\mathcal{U} \subseteq \chi$, $\mathcal{Z} \subseteq \mathcal{Z}_E$, and $\chi \times \mathcal{Z}_E \subseteq \mathcal{L}$, and an extension $\tilde{\Sigma}$ of Σ on $(\mathcal{L}, \leq)$ that is interval compatible with $(\chi, \leq)$ and induced interval compatible with $(\mathcal{L}, \leq)$.*

Proof. To prove that independent discrete state observability implies the existence of a lattice $(\chi, \leq)$ and an extension on $(\mathcal{L}, \leq)$ of Σ that is interval compatible with $(\chi, \leq)$, the reader is deferred to Section 4.1. Briefly, it was shown that the lattice $(\chi, \leq)$ can be chosen as $(\chi, \leq) = (\mathcal{P}(\mathcal{U}), \subseteq)$. The function $\tilde{f} : \chi \times \mathcal{Y} \to \chi$ is defined $\tilde{f}(w, y) = f(\alpha_1, y) \vee ... \vee f(\alpha_n, y)$ for any $w = \alpha_1 \vee ... \vee \alpha_n$, and $\tilde{f}(\perp, y) = \perp$. The function $\tilde{h}$ can be defined on $\chi \times \mathcal{Z}$ as in Section 4.1 so as to guarantee that the $\tilde{\Sigma}$-transition sets defined in Definition 6.3.2 are intervals. We recall that such sets do not depend on z, and thus the same construction developed in Section 4.1 can be repeated. Next, lattices $(\mathcal{Z}_E, \leq)$, and $(\mathcal{L}, \leq)$ with extensions $\tilde{h}$ and $\tilde{F}$ that satisfy the induced interval compatibility properties are constructed as well.

Define $\{z \mid y = g(\alpha, z),\ \alpha \in \mathcal{U}\} := m(\alpha, y)$. Then $\mathcal{Z}_E$ is defined in the following way:

(i) $\mathcal{Z} \subseteq \mathcal{Z}_E$;

(ii) $m(\alpha, y) \in \mathcal{Z}_E$ for any $y \in \mathcal{Y}$ and $\alpha \in \mathcal{U}$;

(iii) $\mathcal{Z}_E$ is invariant under h, i.e., if $\bar{z} \in \mathcal{Z}_E$, then $h(\alpha, \bar{z}) \in \mathcal{Z}_E$ for any $\bar{z} \in \mathcal{Z}_E$ and $\alpha \in \mathcal{U}$;

(iv) $\mathcal{Z}_E$ is closed under finite unions and finite intersections.

By construction, $(\mathcal{Z}_E, \leq)$ is a lattice where the order is established by inclusion. Each element in $\mathcal{Z}_E$ is either a submanifold of $\mathcal{Z}$ or a union of disjoint submanifolds. Also, $(\chi \times \mathcal{Z}_E, \leq)$ is a lattice with order established component-wise. Define $(\mathcal{L}, \leq) := (\mathcal{P}(\chi \times \mathcal{Z}_E), \subseteq)$. Obviously, $\chi \times \mathcal{Z}_E \subseteq \mathcal{L}$. Any element $q \in \mathcal{L}$ has the form $q = (w_1, \bar{z}_1) \vee ... \vee (w_k, \bar{z}_k)$, where $\bar{z}_i \in \mathcal{Z}_E$ and $w_i \in \chi$.

Define the function $\tilde{F} : \mathcal{L} \times \mathcal{Y} \to \mathcal{L}$ in the following way. For any $q = (w_1, \bar{z}_1) \vee ... \vee (w_k, \bar{z}_k) \in \mathcal{L}$, define (we omit the dependence of $\tilde{F}$ on y for simplifying notation)

$$\tilde{F}(q) := \tilde{F}(w_1, \bar{z}_1) \vee ... \vee \tilde{F}(w_k, \bar{z}_n),$$

where

$$\tilde{F}(w_i, \bar{z}_i) := (\tilde{f}(w_i), \tilde{h}(w_i, \bar{z}_i)).$$

Let $w_i = \alpha_{i,1} \vee ... \vee \alpha_{i,p_i}$ and $\bar{z}_i = m_{i,1} \vee ... \vee m_{i,n_i}$ with $m_{i,1}$ submanifolds of $\mathcal{Z}$ or sets of subsets of manifolds of $\mathcal{Z}$, then $\tilde{h} : \chi \times \mathcal{Z}_E \to \mathcal{Z}_E$ is defined such that

$$\tilde{h}(w_i, \bar{z}_i) := \vee_j h(\alpha_{i,j}, \bar{z}_i).$$

From this definition, it follows that $\tilde{F}$ is order preserving. Also, $\tilde{F}(\perp) := \perp$.

The function $\tilde{g} : \mathcal{L} \to \mathcal{Y}$ is defined in the following way. For any $q \in \mathcal{L}$ for $q = (w_1, \bar{z}_1) \vee ... \vee (w_k, \bar{z}_k)$, $w_i = \alpha_{i,1} \vee ... \vee \alpha_{i,p_i}$, and $\bar{z}_i = m_{i,1} \vee ... \vee m_{i,n_i}$

$$\tilde{g}(q) := y \text{ if and only if } \tilde{g}(w_i, \bar{z}_i) = y,$$

with

$$\tilde{g}(w_i, \bar{z}_i) = y \text{ if and only if } g(\alpha_{i,l}, \bar{z}_i) = y \text{ for any } l,$$

where $g(\alpha_{i,l}, \bar{z}_i) = y$ if and only if $\bar{z}_i \subseteq m(\alpha_{i,l}, y)$ by definition of $m(\alpha_{i,l}, y)$.

For any $q = (w_1, \bar{z}_1) \vee ... \vee (w_k, \bar{z}_k) \in \mathcal{L}$, its lower and upper approximations are defined as $a_L(q) := (w_1 \wedge ... \wedge w_k, \bar{z}_1 \wedge ... \wedge \bar{z}_k)$ and $a_U(q) := (w_1 \vee ... \vee w_k, \bar{z}_1 \vee ... \vee \bar{z}_k)$. An example of elements in the lattice $(\mathcal{L}, \leq)$ with their lower and upper approximations is shown in Figure 6.2. The lattices and the system extension have been constructed. Now, the items of

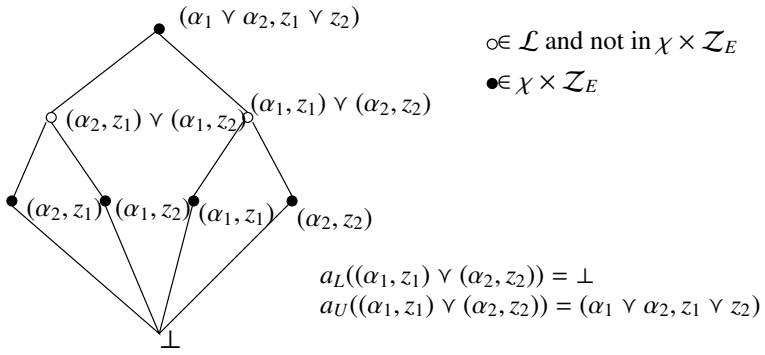


Figure 6.2: Hasse diagram representing elements in the lattice $(\mathcal{L}, \leq)$.

Definition 6.3.6 can be checked. Item (i) of Definition 6.3.6 is satisfied with $\{q \in \mathcal{L} \mid y = \tilde{g}(q), \pi_1 \circ a_L(q) = \perp,\ \pi_1 \circ a_U(q) = w\} = [\perp, u_q(w)]$ with $u_q(kw) = (\alpha_1, m(\alpha_1, y)) \vee ... \vee (\alpha_n, m(\alpha_n, y))$ if $w = \alpha_1 \vee ... \vee \alpha_n$. Also, $a_L(\perp) = \perp$ and $\pi_2 \circ a_U(u_q(w)) = m(\alpha_1, y) \vee ... \vee m(\alpha_n, y)$.

Item (ii) of Definition 6.3.6 is satisfied because $\tilde{F}$ is order preserving by construction and because $\tilde{F} : \alpha \times [\perp, m(\alpha, y)] \to [\perp, \tilde{F}(\alpha, m(\alpha, y))]$ is one-one because the system is observable.

To verify (iii) of Definition 6.3.6, a distance function on $\mathcal{Z}_E$ is defined. For any $\bar{z}_1, \bar{z}_2 \in \mathcal{Z}_E$, define

$$d(\bar{z}_1, \bar{z}_2) := \begin{cases} |\dim(\bar{z}_1) - \dim(\bar{z}_2)| & \text{if } \bar{z}_1 \text{ and } \bar{z}_2 \text{ are related} \\ 1 & \text{if } \bar{z}_1 \text{ and } \bar{z}_2 \text{ are not related,} \end{cases} \tag{6.4}$$

where if $\bar{z} = m_1 \vee ... \vee m_n$, $\dim(\bar{z}) := \sum_i \dim(m_i)$, and $\dim(m_i)$ denotes the dimension of the submanifold $m_i \subset \mathcal{Z}$. Define $\dim(\perp) = 0$, $\dim(z) = 1$ for any $z \in \mathcal{Z}$, thus a submanifold isomorphic to $\mathbb{R}^m$ has dimension $m + 1$. Properties (i)-(ii) of Definition 2.1.4 are verified. (Note that any two points in $\mathcal{Z}$ are not related.) To verify (iii) of the Definition 2.1.4, consider $\bar{z}_1 \leq \bar{z}_2$ for $\bar{z}_1, \bar{z}_2 \in \mathcal{Z}_E$, and compute $d(\perp, \bar{z}_1)$ and $d(\perp, \bar{z}_2)$. If $\bar{z}_1 \leq \bar{z}_2$, by the way $\mathcal{Z}_E$ has been constructed, it means that there are m_i and m'_i submanifolds in $\mathcal{Z}_E$ such that $\bar{z}_1 = m_1 \vee ... \vee m_n$, and $\bar{z}_2 = m'_1 \vee ... \vee m'_p$ with $n \leq p$, and for any i there is a j such that $m_i \subseteq m'_j$. Thus, $\dim(\bar{z}_1) = \dim(m_1) + ... + \dim(m_n)$ and $\dim(\bar{z}_2) = \dim(m'_1) + ... + \dim(m'_p)$

with $n \leq p$ and $\dim(m_i) \leq \dim(m'_i)$. Thus expression (6.4) defines a distance function according to Definition 2.1.4. Thus, for any $[\perp, U] \subseteq \chi$ with $U = \alpha_1 \vee ... \vee \alpha_n$, we have that

$$d(\perp, \pi_2 \circ a_U \circ \tilde{F}(u_q(U))) = d(\perp, h(\alpha_1, m(\alpha_1, y)) \vee ... \vee h(\alpha_n, m(\alpha_n, y))),$$

as

$$\tilde{F}(u_q(U)) = (f(\alpha_1), h(\alpha_1, m(\alpha_1, y)) \vee ... \vee (f(\alpha_n), h(\alpha_n, m(\alpha_n, y)),$$

and

$$a_U \circ \tilde{F}(u_q(U)) = (f(\alpha_1) \vee ... \vee f(\alpha_n), h(\alpha_1, m(\alpha_1, y)) \vee ... \vee h(\alpha_n, m(\alpha_n, y))),$$

and thus

$$\pi_2 \circ a_U \circ \tilde{F}(u_q(U)) = h(\alpha_1, m(\alpha_1, y)) \vee ... \vee h(\alpha_n, m(\alpha_n, y)).$$

Concluding, the definition of distance given in equation (6.4) yields to

$$d(\perp, h(\alpha_1, m(\alpha_1, y)) \vee ... \vee h(\alpha_n, m(\alpha_n, y))) = \sum_{i=1}^{n} \dim(h(\alpha_i, m(\alpha_i, y)) \leq d_M |[\perp, U]|,$$

where $d_M = \max_i \dim(h(\alpha_i, m(\alpha_i, y))$. □

This theorem shows that for observable and independent discrete state observable systems it is always possible to construct the estimator on a lattice proposed in Theorem 6.3.1. However, the main advantage of using the partial order based approach to state estimation is clear from a computational complexity standpoint when the space of discrete and/or the space of continuous variables can be extended to lattices where the order relation can be efficiently computed. A class of systems for which this is the case is shown in the next section.

6.5 The Case of Monotone Systems

In this section, we show a class of systems in which there is a partial order on $\mathcal{Z}$, the cone partial order, that is preserved by the system dynamics. In this case $\mathcal{Z}_E = \mathcal{Z}$, $(\mathcal{Z}, \leq)$ is a lattice, and $(\mathcal{L}, \leq) = (\chi \times \mathcal{Z}, \leq)$, i.e., $(\mathcal{L}, \leq)$ is the Cartesian product of the partial orders on the spaces of discrete and continuous variables.

Monotone dynamical systems are usually defined on ordered Banach spaces, which we now introduce.

Definition 6.5.1. (Ordered Banach space) An *ordered Banach space* is a real Banach space $\mathcal{Z}$ with a non-empty closed subset K known as the positive cone with the following properties:

(i) $\alpha K \subseteq K$ for any $\alpha \in \mathbb{R}_+$;

(ii) $K + K \subseteq K$;

(iii) $K \cap (-K) = \{\emptyset\}$, i.e., the cone is pointed.

A partial ordering is then defined by $x \geq y$ for any $x, y \in \mathcal{Z}$ if and only if $x - y \in K$, with $x > y$ if and only if $x \geq y$ and $x \neq y$. The space and the partial order is denoted $(\mathcal{Z}, \leq)$.

For more details on ordered Banach spaces and monotone systems, the reader is deferred to [40] and [8].

From now on, let $(\mathcal{Z}, \leq)$ be an ordered Banach space. A monotone dynamical system on $(\mathcal{Z}, \leq)$ is one whose flow preserves the ordering on initial data. To extend this property to deterministic transition systems, consider the extension of $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$ on the lattice $(\chi \times \mathcal{Z}, \leq)$. Such extension, denoted $\tilde{\Sigma} = (\chi \times \mathcal{Z}, \mathcal{Y}, (\tilde{f}, \tilde{h}), \tilde{g})$, is by definition such that $\tilde{f} : \chi \times \mathcal{Y} \to \chi$ and $\tilde{f}|_{\mathcal{U}\times\mathcal{Y}} = f$; $\tilde{h} : \chi \times \mathcal{Z} \to \mathcal{Z}$ with $\tilde{h}|_{\mathcal{U}\times\mathcal{Z}} = h$; $\tilde{g} : \chi \times \mathcal{Z} \to \mathcal{Y}$ and $\tilde{g}|_{\mathcal{U}\times\mathcal{Z}} = g$. The only portion of the space that in fact has been extended is the discrete portion as the continuous portion is an ordered Banach space already.

Definition 6.5.2. (Monotone deterministic transition systems) A deterministic transition system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$, with $(\mathcal{Z}, \leq)$ an ordered Banach space and $(\chi, \leq)$ a lattice

with $\mathcal{U} \subseteq \chi$, is said to be *a monotone deterministic transition system* on the partial order $(\chi \times \mathcal{Z}, \leq)$ if there is an extension $\tilde{\Sigma} = (\chi \times \mathcal{Z}, \mathcal{Y}, (\tilde{f}, \tilde{h}), \tilde{g})$ on $(\chi \times \mathcal{Z}, \leq)$ with the property that $\tilde{h} : \chi \times \mathcal{Z} \to \mathcal{Z}$ is order preserving. The extension $\tilde{\Sigma}$ is termed the *monotone extension* of Σ on $(\chi \times \mathcal{Z}, \leq)$.

For a monotone deterministic transition system, the partial order $(\mathcal{Z}, \leq)$ can be used in the estimator design to bring the computational burden down, as the elements of $\mathcal{Z}$ are points, and their partial order relation can be computed efficiently using the algebraic definition of $(\mathcal{Z}, \leq)$. This avoids the complexity of the representation of elements such as the ones in the constructive proof of Theorem 6.4.1, in which the elements in $\mathcal{Z}_E$ are sets of points of $\mathcal{Z}$, specifically manifolds, intersection of manifolds, and union of manifolds.

6.5.1 Form of the Estimator for a Monotone System

For a monotone deterministic transition system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{Y}, (f, h), g)$, the induced transition sets take a new form. Consider the monotone extension $\tilde{\Sigma} = (\chi \times \mathcal{Z}, \mathcal{Y}, (\tilde{f}, \tilde{h}), \tilde{g})$ of Σ on $\chi \times \mathcal{Z}$ and a transition set $T_{(y_1,y_2)}(\tilde{\Sigma})$ for some $y_1, y_2 \in \mathcal{Y}$. For any $w_1, w_2 \in T_{(y_1,y_2)}(\tilde{\Sigma})$ with $w_1 \leq w_2$, the induced transition sets have the form

$$T^{(w_1,w_2)}_{(y_1,y_2)}(\tilde{\Sigma}) = \{z \in \mathcal{Z} \mid y_2 = \tilde{g}(\tilde{f}(w, y_1), \tilde{h}(w, z)) \text{ and } y_1 = \tilde{g}(w, z)\},$$

in which now $T^{(w_1,w_2)}_{(y_1,y_2)}(\tilde{\Sigma}) \subseteq \mathcal{Z}$. As a consequence, also the induced interval compatibility takes a new form. In particular, the induced transition sets must be intervals in $\mathcal{Z}$ according to the cone order established in $\mathcal{Z}$. More formally, we can redefine the induced interval compatibility as follows.

Definition 6.5.3. (Induced interval compatibility-monotone case) The pair $(\tilde{\Sigma}, (\mathcal{Z}, \leq))$ is said to be *induced interval compatible* if for any $w_1, w_2 \in T_{y_1,y_2}(\tilde{\Sigma})$ for some $y_1, y_2 \in \mathcal{Y}$ with $w_1 \leq w_2$, we have that

(i) the induced transition sets are such that

$$\begin{aligned} \bigwedge T^{(w_1,w_2)}_{(y_1,y_2)} &= l_z(w_1) \\ \bigvee T^{(w_1,w_2)}_{(y_1,y_2)} &= u_z(w_2); \end{aligned}$$

(ii) $\tilde{h} : \alpha \times [l_z(\alpha), u_z(\alpha)] \to [\tilde{h}(\alpha, l_z(\alpha)), \tilde{h}(\alpha, u_z(\alpha))]$ is order isomorphic for any $\alpha \in \mathcal{U}$;

(iii) $d(\tilde{h}(w_1, l_z(w_1)), \tilde{h}(w_2, u_z(w_2))) \leq \gamma(|[w_1, w_2]|)$, with "$d$" a distance function on $\mathcal{Z}$, and with $\gamma : \mathbb{N} \to \mathbb{R}$ a monotonic function of its argument.

Then, the form of the estimator of Theorem 6.3.1 is given by the same f_1 and f_2, and by $f_3 : \mathcal{Z} \times \chi \times \mathcal{Y} \times \mathcal{Y} \to \mathcal{Z}$, $f_4 : \mathcal{Z} \times \chi \times \mathcal{Y} \times \mathcal{Y} \to \mathcal{Z}$, $f_5 : \chi \to \chi$ with $f_5 = \text{id}$, $f_6 : \mathcal{Z} \to \mathcal{Z}$ with $f_6 = \text{id}$, $f_7 : \mathcal{Z} \to \mathcal{Z}$ with $f_7 = \text{id}$ defined as

$$\begin{aligned} f_3(z_L(k), L(k), y(k), y(k+1)) &= \tilde{h}(\textstyle\bigwedge O_y(k) \vee L(k), z_L(k) \vee l_z(\bigwedge O_y(k) \vee L(k))) \\ f_4(z_U(k), U(k), y(k), y(k+1)) &= \tilde{h}(\textstyle\bigvee O_y(k) \wedge U(k), z_U(k) \wedge u_z(\bigvee O_y(k) \wedge U(k))), \\ f_6(z_L(k)) &= \text{id} \\ f_7(z_U(k)) &= \text{id}. \end{aligned}$$

The results of the Theorem 6.3.1 remain the same except for property (iii') that changes to

(iii') there exists $k_0' > 0$ such that for any $k \geq k_0'$, $d(z_{L'}(k), z_{U'}(k)) = 0$ where

$$\begin{aligned} L'(k) &= \textstyle\bigwedge([L(k), U(k)] \cap \mathcal{U}) \\ U'(k) &= \textstyle\bigvee([L(k), U(k)] \cap \mathcal{U}) \\ z_{L'}(k+1) &= f_3(z_{L'}(k), L'(k), y(k), y(k+1)) \\ z_{U'}(k+1) &= f_4(z_{U'}(k), U'(k), y(k), y(k+1)), \end{aligned}$$

with $z_{L'}(0) = z_L(0)$ and $z_{U'}(0) = z_U(0)$.

The schematic of Figure 6.1 transforms to the one in Figure 6.3.

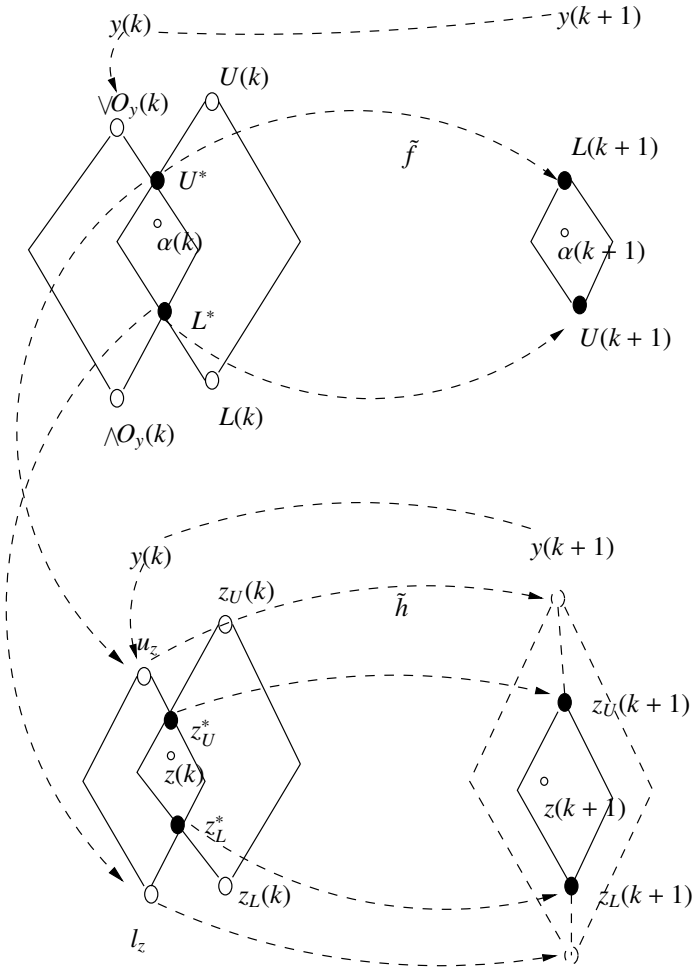


Figure 6.3: Hasse diagrams representing the updates of the estimator in Theorem 6.3.1 for the case of monotone systems. In the diagram, we have denoted $U^* = \vee O_y(k) \wedge U(k)$, $L^* = \wedge O_y(k) \vee L(k)$, $z^*_U = z_U(k) \wedge l_z(U^*)$, and $z^*_L = z_L(k) \vee l_z(L^*)$.

Corollary 6.5.1. *If in addition to the assumptions of Theorem 6.3.1, $\tilde{\Sigma}$ is observable and independent discrete state observable, then we have the stronger convergence properties:*

(iv) there exists $k'_0 > 0$ such that for any $k \geq k'_0$ $d(z_L(k), z_U(k)) = 0$;

(v) there exist a $k_0 > 0$ such that for any $k > k_0$ $L(k) = U(k) = \alpha(k)$.

For the proof of (v), the reader is deferred to Section 3.3. The proof of (iv) can be carried out by contradiction in a way analogous to how (iii') of Theorem 6.3.1 was proved.

6.5.2 Algebraic Tests for Induced Interval Compatibility

In the case of monotone systems, an algebraic check can be performed to verify the interval compatibility properties once a lattice $(\chi, \leq)$ is chosen for the discrete state space. Define

$$\tilde{h}^k(w, z) := \tilde{h}(\tilde{h}^{k-1}(w, z), \tilde{f}^{k-1}(w, y(k-2))),$$

and

$$\tilde{f}^k(w, y(k-1)) := \tilde{f}(\tilde{f}^{k-1}(w, y(k-2), y(k-1)),$$

with $\tilde{f}^0(w, y) := w$ and $\tilde{h}^0(w, z) := z$. The following proposition is a straightforward consequence of the observability property of a system.

Proposition 6.5.1. *Consider the monotone deterministic transition system* $\Sigma = (\mathcal{U}\times\mathcal{Z}, \mathcal{Y}, (f, h), g)$. *If its monotone extension* $\tilde{\Sigma}$ *is observable, there is* $\bar{k} > 0$ *such that*

$$\{z \mid \tilde{g}(w_0, z) = y(0), ..., \tilde{g}(\tilde{h}^{\bar{k}-1}(w_0, z), \tilde{f}^{\bar{k}-1}(w_0, y(\bar{k}-2)) = y(\bar{k}-1)\} = \{z(0)\},$$

where $y(k) = \tilde{g}(w(k), z(k))$, *and* $w_0 = w(0)$.

This proposition indicates that if the system $\tilde{\Sigma}$ is observable, the continuous state z can be expressed as a function of the output sequence and of the starting discrete state. Thus, there is a map that attaches to a discrete state, a value of the continuous state after some time given an output sequence: this map is defined to be the observability map. In general, $\bar{k}$ depends on z. In case it is not dependent on z, we say that the system is observable in $\bar{k}$ steps.

Definition 6.5.4. (Observability map) Let the monotone extension $\tilde{\Sigma}$ of Σ be observable. Let $y = \{y(k)\}_{k\in[1,\bar{k}]}$ be the output sequence up to the smallest step $\bar{k}$ such that the system of equations

$$\begin{aligned} \tilde{g}(z, w) &= y(0) \\ &\vdots \\ \tilde{g}(\tilde{h}^{\bar{k}-1}(w, z), \tilde{f}^{\bar{k}-1}(w, y(\bar{k}-2))) &= y(\bar{k}-1) \end{aligned}$$

has a unique solution for $z \in \mathcal{Z}$. Then, the *observability map*, denoted $O_y : \chi \to \mathcal{Z}$, is the map that for a fixed finite sequence y attaches to w the unique z satisfying the above system.

Then, we can give the algebraic condition that guarantees that $\tilde{\Sigma}$ is induced interval compatible with $(\mathcal{Z}, \leq)$.

Proposition 6.5.2. *If the monotone extension of* Σ, $\tilde{\Sigma}$ *is observable in two steps, and the observability map* $O_y : \chi \to \mathcal{Z}$ *is order preserving, then the pair* $(\tilde{\Sigma}, (\mathcal{Z}, \leq))$ *is induced interval compatible.*

Proof. To prove (i) of Definition 6.3.6, let $y = (y(k), y(k+1))$ be a pair of consecutive outputs in the output sequence $\{y(k)\}_{k\in\mathbb{N}}$ corresponding to an execution of $\tilde{\Sigma}$. By the observability in two steps hypothesis, it follows that for a fixed $w \in \chi$

$$\{z \in \mathcal{Z} \mid y(k) = \tilde{g}(w, z),\ y(k+1) = \tilde{g}(\tilde{h}(w, z), \tilde{f}(w, y(k)))\} = \{z^*\},$$

and thus $l_z(w) = z^* = u_z(w)$. Also, by the Definition 6.5.4, it follows that $z^* = O_y(w)$. By the order preserving property of O_y, it follows that $O_y(w_1) \leq O_y(w_2)$ if $w_1 \leq w_2$. Item (ii) of Definition 6.3.6 is clearly verified as $l_z(\alpha) = u_z(\alpha)$. Item (iii) can be proved in the following way. Let

$$\bar{d} := \max_{w_i \ll w_j} \|\tilde{h}(w_i, O_y(w_i)) - \tilde{h}(w_j, O_y(w_j))\|$$

for $w_i, w_j \in [w_1, w_2] \subseteq \chi$, then (iii) is verified with $\gamma(|[w_1, w_2]|) = \bar{d}|[w_1, w_2]|$. □

As a consequence of this proposition, the check for induced interval compatibility is the order preserving property of the output map O_y, which is easy to check. The basic assumption in order to have induced interval compatibility, is the order preserving property of the observability map. In fact, the two steps observability assumption can be abolished if item (i) of Definition 6.3.6 is relaxed to consider a longer sequence of output observations. This can be done with minor modifications.

6.6 Simulation Examples

The first example is a linear hybrid automaton, in which a lattice of the type constructed in the proof of Theorem 6.4.1 is used. The second example is a monotone deterministic transition system in which the discrete space lattice is constructed as in the proof of Theorem 6.4.1. This allows to have $\mathcal{Z}_E = \mathcal{Z}$ with a cone partial order with some complexity reduction. However, the discrete space lattice still has the worst case size, and its partial order relation needs to be stored. The third example is the multi-robot example already proposed in Section 3.4, in which now the defenders have second order dynamics and only their positions are measured. This is a monotone deterministic transition system in which also the discrete state has been extended to a lattice whose partial order relations can be efficiently computed using algebraic properties. This is the case that allows the largest complexity reduction. This section is then concluded with complexity computations for each one of the three examples proposed.

6.6.1 Example 1: Linear Discrete-Time Hybrid Automaton

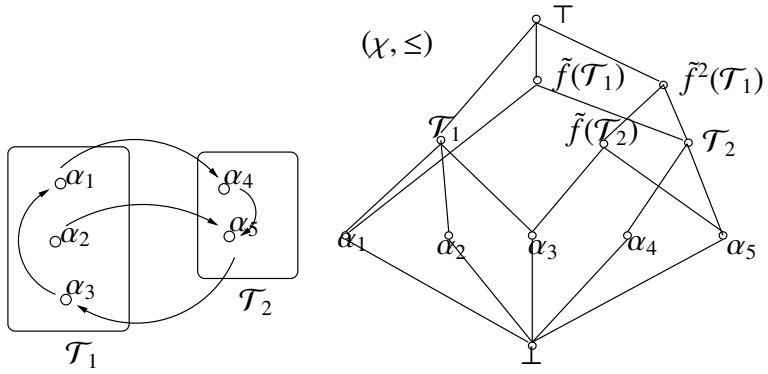


Figure 6.4: Map f and output function for the automaton of Example 1 (left). Lattice $(\chi, \leq)$ and the extended function $\tilde{f}$ (right).

Let $\mathcal{U} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$, and $\alpha(k+1) = f(\alpha(k))$ where f is defined in the Figure 6.4 (left). Assume $\mathcal{Z} = \mathbb{R}^n$, $z(k+1) = A(\alpha(k))z(k) + B(\alpha(k))$, where $A(\alpha_i) = A_i \in \mathbb{R}^n \times \mathbb{R}^n$ and $B(\alpha_i) = B_i \in \mathbb{R}^n$. The output function g is such that $g(\alpha, z) = (g_\alpha(\alpha), g_z(\alpha, z))$, where $g_\alpha : \mathcal{U} \to \{\mathcal{T}_1, \mathcal{T}_2\}$ and $g_z(\alpha, z) = C(\alpha)z$, with $C(\alpha_i) = C_i \in \mathbb{R}^m \times \mathbb{R}^n$.

An instance of such an example is considered with $n = 3$, where $A_1 = ((1,1,1)', (0,1,1)', (0,0,1)')'$, $A_2 = ((1/2,1/2,1/2)', (1,2,2)', (0,0,1)')'$, $A_3 = ((2,1,1)', (0,1,1)', (2,0,0)')'$, $A_4 = ((1,1,1)', (1,1,0)', (0,0,1)')'$, $A_5 = ((1,0,0)', (1,1,1)', (1,1,0)')'$, $C_1 = (1,0,0)$, $C_2 = (1,1,2)$, $C_3 = (0,0,0)$, $C_4 = (1,0,0)$, and $C_5 = (0,1,1)$. The values of B_i are not relevant for computing the estimator performance, and thus they are omitted.

For the discrete state estimate, the minimal lattice $(\chi, \leq)$ where the system is extended is shown in Figure 6.4 right. Its size is always smaller than $|\mathcal{U}|^2$ as shown in Proposition 4.1.2.

For the continuous state estimate, the lattice $(\mathcal{Z}_E, \leq)$ is constructed according to the proof of Theorem 6.4.1, where the submanifolds are affine linear subspaces. Thus, $z_U(k)$ at each step k is a collection of affine linear subspaces, each given by the set of $z \in \mathbb{R}^3$ such that $M_i(k)z = (Y(k) - V_i(k))$, where $M_i(k) = (C(\alpha_i)', (C(f(\alpha_i))A(\alpha_i))', ..., (C(f^{k-1}(\alpha_i))A(f^{k-2}(\alpha_i)))')'$, $V_i(k) = (0, C(f(\alpha_i))B(\alpha_i), ...C(f^{k-1}(\alpha_i))B(f^{k-2}(\alpha_i)))'$, $Y(k) = (y(0), ..., y(k-1))'$, and α_i is such that $f^{k-1}(\alpha_i) \in [\perp, U(k)]$, for $U(k) \in \chi$ and $i \in \{1, .., 5\}$. When only one α_i is left in $[\perp, U(k)]$ and the corresponding matrix $M_i(k)$ has rank equal to n, the estimator has converged. Thus, define $d(\perp, z_U(k)) = \sum_{i=1}^{5} \beta(M_i(k))$ where

$$\beta(M_i(k)) := \begin{cases} 0 & \text{if } f^{k-1}(\alpha_i) \notin [\perp, U(k)] \\ (n+1) - \text{rank}(M_i(k)) & \text{otherwise.} \end{cases}$$

As a consequence, when $d(\perp, z_U(k)) = 1$, the estimator has converged and $z(k) = M_j(k)^\dagger (Y(k) - V_j(k))$ for some $j \in \{1, ..., 5\}$, where $M_j(k)^\dagger$ is the pseudoinverse of $M_j(k)$. Note that, after the first k at which $d(\perp, z_U(k)) = 1$, the state of the system is tracked. The behavior of $d(\perp, U(k)) := |[\perp, U(k)]|$ and of $d(\perp, z_U(k))$ are illustrated in the left plot of Figure 6.5. Note that the simultaneous discrete-continuous state estimation allows faster convergence rates of the continuous estimate with respect to the case in which the continuous estimate would take place after the discrete estimate has converged.

In this example, the continuous variable space does not have monotone properties. As a consequence, the representation of the elements of $(\chi, \leq)$ and of $(\mathcal{Z}_E, \leq)$ involves a listing of objects: for χ, there is a listing of α_is and for $\mathcal{Z}$ we have a listing of linear subspaces.

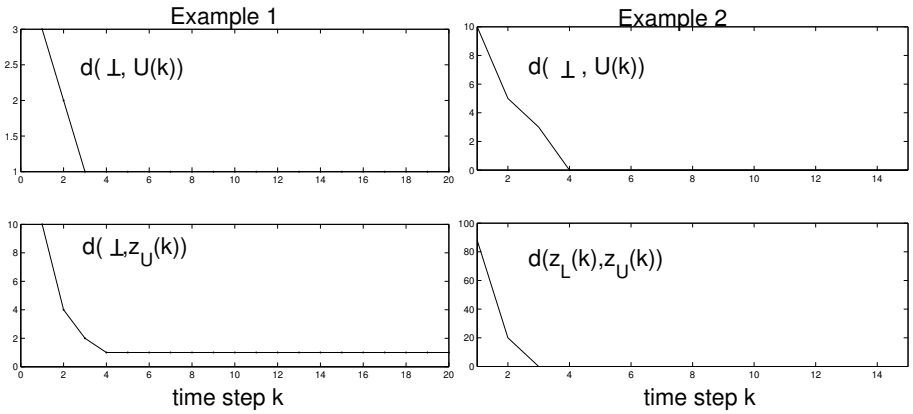


Figure 6.5: Estimator performance: example 1 (left) and example 2 (right).

Moreover, to represent each linear subspace, a number of constants larger than n (the number of constants needed for representing an element in $\mathbb{R}^n$) is needed. A measure of the complexity of the estimator is given in the sequel. If $|\mathcal{U}|$ is very large, this choice of the partial orders renders the estimation process prohibitive. A case in which a different partial order must be used for computational tractability, is presented in Example 3.

6.6.2 Example 2: Monotone System

This example considers the case in which it is possible to choose $\mathcal{Z}_E = \mathcal{Z}$ because the system is monotone. Let again $\mathcal{U} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$, and $\alpha(k+1) = f(\alpha(k))$ where f is defined in Figure 6.4 (left). The continuous dynamics is given by

$$\begin{aligned} z_1(k+1) &= (1-\beta)z_1(k) - \beta z_2(k) + 2\beta X(\alpha(k)) \\ z_2(k+1) &= (1-\lambda)z_2(k) + \lambda X(\alpha(k)), \end{aligned} \tag{6.5}$$

where $\beta = 0.1$, $\lambda = 0.1$, $X(\alpha_i) := 10i$ for $i \in \{1, ..., 5\}$. The minimal lattice $(\chi, \leq)$ is shown in Figure 6.4 (right). In this case $\mathcal{L} = \chi \times \mathcal{Z}$, where $\mathcal{Z} = \mathbb{R}^2$, and the order $(\mathcal{Z}, \leq)$ is chosen such that $(z_1^a, z_2^a) \leq (z_1^b, z_2^b)$ if and only if $z_2^a \leq z_2^b$. The function $\tilde{h} : \chi \times \mathcal{Z} \to \mathcal{Z}$ is defined by defining the function $\tilde{X} : \chi \to \mathbb{R}$ in the following way. $\tilde{X}(\mathcal{T}_1) := \max(X(\alpha_1), X(\alpha_2), X(\alpha_3)) = 30$, $\tilde{X}(\mathcal{T}_2) := \max(X(\alpha_3), X(\alpha_5)) = 50$, and in an analogous way for the others, that is $\tilde{X}(\tilde{f}(\mathcal{T}_2)) = 50$, $\tilde{X}(\tilde{f}^2(\mathcal{T}_1)) = 50$, $\tilde{X}(\tilde{f}(\mathcal{T}_1)) = 50$, and $\tilde{X}(\perp) := 0$.

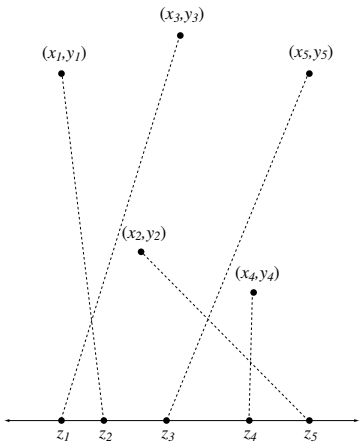


Figure 6.6: An example state of the RoboFlag Drill for 5 robots. Here $\alpha = \{3, 1, 5, 4, 2\}$.

With this choice, $\tilde{h}(w_1, z^a) \leq \tilde{h}(w_2, z^b)$ for any $(w_1, z^a) \leq (w_2, z^b)$, that is, the system is monotone. Convergence plots are shown in Figure 6.5 (right).

As opposite to Example 1, in this case the representation of the elements in $\mathcal{Z}_E$ requires only n scalar numbers, and the computation of the order relation is straightforward. This alleviates the computational burden with respect to the previous example.

6.6.3 Example 3: RoboFlag Drill (variation)

A version of the RoboFlag Drill system, already presented in Section 3.4, is considered in which now the robots have partially measured second order dynamics. Briefly, there are two teams of N robots, say the attackers and the defenders, in which each defender is assigned to an attacker and moves toward it in order to intercept it before it passes over a defensive zone. There is an assignment protocol that establishes that two close defenders moving one toward the other will exchange their assignments. Only the dynamics of the defenders is different from Section 3.4. In this case in fact, they have second order dynamics in which the state is not entirely measured. Figure 6.6, represents an example with five robots per team. The attacker positions are denoted by (x_i, y_i) and their dynamics is given by

$$\text{if } y_i > \delta \text{ then } y_i' = y_i - \delta.$$

For the defenders, let the assignment be denoted by $\alpha = (\alpha_1, ..., \alpha_N) \in \text{perm}(N)$, with α_i the assignment of defender i, $\mathcal{U} = \text{perm}(N)$, their state variable be denoted by $z = (z_{1,1}, z_{1,2}, ..., z_{N,1}, z_{N,2}) \in \mathcal{Z}$, with output $(z_{1,1}, ..., z_{N,1}) \in \mathcal{Y}$. The function $f : \mathcal{U} \times \mathcal{Y} \to \mathcal{U}$ that updates α is given by

$$\text{if } x_{\alpha_i} < z_{i,1} \text{ and } x_{\alpha_{i+1}} < z_{i+1,1} \text{ then } (\alpha'_i, \alpha'_{i+1}) = (\alpha_{i+1}, \alpha_i), \tag{6.6}$$

for any i. The function $h : \mathcal{U} \times \mathcal{Z} \to \mathcal{Z}$ that updates the z variables is given by

$$\begin{aligned} z'_{i,1} &= (1-\beta)z_{i,1} - \beta z_{i,2} + 2\beta x_{\alpha_i} \\ z'_{i,2} &= (1-\lambda)z_{i,2} + \lambda x_{\alpha_i} \end{aligned} \tag{6.7}$$

for any i. The set $\mathcal{Z}$ is such that $z_{i,1} \in [x_i, x_{i+1}]$ and $z_{i,2} \in [x_i, x_{i+1}]$ for any i, which is guaranteed if β and λ are assumed sufficiently small.

It can be easily shown that the system is independent discrete state observable and interval compatible with $(\chi, \leq)$ defined in the following way. The set χ is the set of vectors in $\mathbb{N}^N$ with components less than N, and the order between any two vectors in χ is established component-wise. By construction $\text{perm}(N) \subset \chi$. It can be verified that the extended system is observable in two steps. Also, we have the following property.

Proposition 6.6.1. *The system Σ reported in equations (6.6) and (6.7) is monotone, and the output map is order preserving.*

Proof. We show that the system is monotone, by showing that there is a positive cone in $\mathcal{Z}$ that induces the partial order $(\mathcal{Z}, \leq)$, and a lattice $(\chi, \leq)$ such that the extended system $\tilde{\Sigma}$ on $(\chi, \leq)$ is as in Definition 6.5.2. Let us choose $(\chi, \leq)$ to be the set of vectors in $\mathbb{N}^N$ with components less than N, with the order between any two vectors in χ established component-wise. For showing that $\tilde{h} : \chi \times \mathcal{Z} \to \mathcal{Z}$ is order preserving, we choose the positive cone K in $\mathcal{Z}$ composed by all vectors $v = (v_{1,1}, v_{1,2}, ..., v_{N,1}, v_{N,2})$ such that $v_{i,2} \geq 0$. This basically means that the order on each $z_{i,2}$ must be preserved by the dynamics in equations (6.7). This is true as if $z^{(1)}_{i,2} < z^{(2)}_{i,2}$ and $w^{(1)}_i \leq w^{(2)}_i$ then $(1-\lambda)z^{(1)}_{i,2} + \lambda x_{w^{(1)}_i} \leq (1-\lambda)z^{(2)}_{i,2} + \lambda x_{w^{(2)}_i}$ because $x_{w^{(1)}_i} \leq x_{w^{(2)}_i}$ whenever $w^{(1)}_i \leq w^{(2)}_i$, and because $(1-\lambda) > 0$.

The output map is readily seen to be order preserving in its argument $w = (w_1, ..., w_N) \in \chi$ as for any k, we have that $z_{i,2}(k) = \frac{1}{\beta}\left((1-\beta)y_i(k) - y_i(k+1) + 2\beta x_{w_i(k)}\right)$. □

The estimator in equations (6.4) has been implemented for the system in equations (6.6) and (6.7). The discrete state estimator is identical to the one in Section 3.4. For the continuous state estimator set $z_L = (z_{L,1}, ..., z_{L,N}) \in \mathbb{R}^N$ and $z_U = (z_{U,1}, ..., z_{U,N}) \in \mathbb{R}^N$, where $z_{L,i} \leq z_{i,2} \leq z_{U,i}$, that is, $z_{L,i}$ and $z_{U,i}$ are the lower and upper bound of the $z_{i,2}$, respectively. The first components $z_{i,1}$ are neglected as they are measured. Figure 6.7 illustrates the estimator performance. $W(k) = \sum_{i=1}^{N} |m_i(k)|$, where $|m_i(k)|$ is the cardinality of the sets

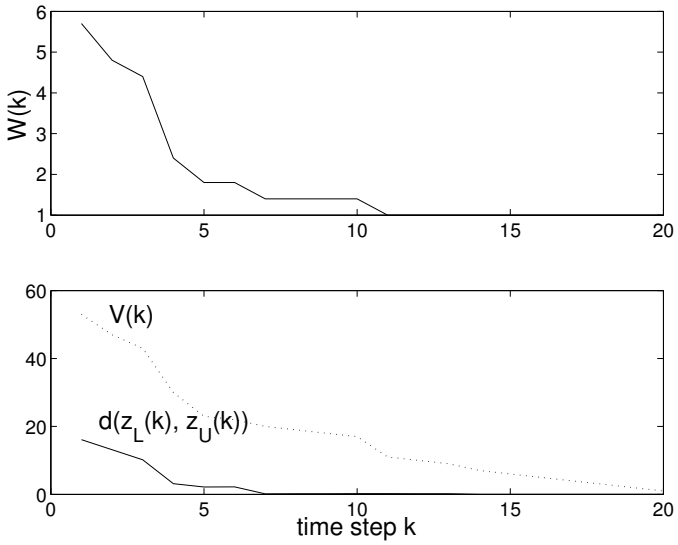


Figure 6.7: Estimator performance with $N = 10$ agents.

$m_i(k)$ that are the sets of possible α_i for each component obtained from the sets $[L_i, U_i]$ by removing iteratively a singleton occurring at component i by all other components. When $[L(k), U(k)] \cap \text{perm}(N)$ has converged to α, then $m_i(k) = \alpha_i(k)$ (see Section 3.4 for details). The distance function for $z, x \in \mathbb{R}^N$ is defined

$$d(x, z) = \sum_{i=1}^{N} \text{abs}(z_i - x_i).$$

The function $V(k)$ is defined as

$$V(k) = \frac{1}{2}\sum_{i=1}^{N}(x_{U_i(k)} - x_{L_i(k)}),$$

and it is always non increasing. Note that even if the discrete state has not converged yet, the continuous state estimation error after $k = 8$ is close to zero.

6.6.4 Complexity Considerations

The scope of the proposed examples is two-fold. First, they give an idea of the range of systems to which the lattice estimation approach applies (observable and independent discrete state observable systems). Second, they point out that the lattice approach alleviates the computational burden of the estimator and even renders intractable problems tractable when the system has monotone properties and a good choice of the lattices is made. To make this point more formal, the computational complexity in each of the examples is estimated as a function of the continuous variables, the discrete variables, and the sizes of the sets where the discrete variables lie. This section is not meant to be a formal treatment of computational complexity, but has the scope of giving a qualitative measure of the computational complexity diversity of the proposed examples. Let n be the number of continuous variables (3 for the first example, 2 for the second, and 20 in the third), N be the number of discrete variables (1 in the first example, 1 in the second example, and 10 in the third example), and u be the set where each discrete variable lies (in the first and second example $u = \mathcal{U}$, and in the third $u = \{1, ..., N\}$ and $\mathcal{U} = u^N$). The computational cost of the estimator is computed as

$$\text{computational cost} \propto S + a_{UC}$$

where S is the sum of the sizes of the look-up tables used at each update of the estimator, and a_{UC} is the algebraic update cost of each estimator update. The cost of any set of algebraic computation is set to 1. One can verify that $S \propto |u|^{2N}$ in the first two examples, and that $S \propto 2N$ in the third one. In the first example, $a_{UC} \propto |u|^N n$, and $a_{UC} \propto 2n$ in the second and third examples. This is shown in the following table.

Estimator computational cost

Example 1	$\|u\|^{2N} + \|u\|^{N} n$
Example 2	$\|u\|^{2N} + 2n$
Example 3	$2N + 2n$

From the table, one notices that moving from Example 1 to Example 3 the computational burden due to the size of u decreases, and it disappears in the case of the third example. This is due to the monotone properties of the continuous dynamics in Example 2 and Example 3, and to the existence of a lattice $(\chi, \leq)$ with algebraic properties in Example 3. Note also that the complexity reduction that characterizes the third example does not occur because the discrete variables dynamics decouples, as in fact it is heavily coupled.

In order to give an idea of how one can find a "good" partial order for reducing the complexity of the estimator design as it happens in Example 3, we consider in the next chapter some application domains for which we show how a good partial order can be established. The main idea for a general system is to find a coordinate frame in which the system evolves preserving some partial order.

Chapter 7

Conclusions, Future Directions, and Possible Extensions

7.1 Conclusions

In this work, we have presented an approach to state estimation in decision and control systems, which reduces complexity by using partial order theory. The main idea is the one of representing sets of consistent states by a lower and an upper bound once a partial order has been established on the set of states. Under order preserving assumptions on the system's dynamics, the estimator can just keep track of the lower and upper bounds of the set of all consistent states. Under suitable observability assumptions, it has been shown that the lower and upper bounds converge toward each other. The main advantage of this approach with respect to enumeration approaches most often used in the literature is that a "cheap" representation of the sets of interest can be used in the estimator. This synthetic representation and the fact that the representation is preserved by the dynamics of the system guarantees that the computational burden is drastically reduced with respect to enumeration approaches [13, 14, 22].

The generality of this approach has been investigated. In particular, it was shown that if the system is observable, one can always find a partial order that allows the construction of the estimator. The size and the complexity of the representation of the elements of the partial order determine the complexity of the estimator. In the worst case scenario, one can always find a partial order that leads to an estimator whose computation burden is the same

as the computation burden of the enumeration approach. In addition, it was shown how the ideas developed in the context of deterministic systems can be generalized to systems that are affected by uncertainty.

The developed algorithms enjoy scalability properties that are substantial in multi-agent systems. This has been done for estimating the discrete state once the continuous one is measured and for estimating both discrete and continuous state when an estimator in cascade form is possible. The question of how to deal with the estimation problem for both the continuous and the discrete states when an estimator cannot be put in cascade form is still to be addressed. In particular, we will consider this question by requiring a bound on the computational burden needed for implementing the estimator. With this bound, we conjecture that with the partial order approach to state estimation it will be possible to develop state estimators with low computational burden to the expense of estimation accuracy. This is a compromise between performance and complexity.

For the problem of estimating the discrete state, we have shown that the size of the lattice chosen for the estimator construction affects complexity. It is interesting to explore if a minimal lattice always exists, what is the complexity required for its computation, and if there is an automatic procedure to construct it. In the context of estimation of both the continuous and the discrete states, we would like to explore possible analogies between the partial order that is preserved by the dynamics and Lyapuniov functions for dynamical systems. This would be useful to clarify any physical meaning that a partial order preserved by the system dynamics may have.

By the use of a partial order, we were able to reason about notions such as convergence, stability, and performance in a discrete space in a way similar to how we reason about these notions in a continuous state space. This fact led us to overcome the dichotomy between the discrete world and the continuous world, which affects virtually all state estimation algorithms for hybrid systems proposed in the literature [2, 3, 4, 5]. Partial order theory has proved to be a useful tool borrowed from theoretical computer science to address this issue, and it was nicely merged with classical control theory to reach our goal. Using partial order theory, can we build a bridge between the continuous and the discrete world for dealing with more general analysis and control problems as well? This is the subject of

current and future work.

In the next section, we give some hints on possible application areas and related extensions of the proposed estimation approach. In particular, we show how to use the state estimation algorithms on a lattice to reduce computational complexity when solving a monitoring problem of distributed environments. The key point for applying our estimator in a way such that complexity is reduced is the one of finding a good coordinate frame for describing the system. Once this is done, the theory applies with minor extensions. Simulation examples show promising performance.

7.2 Future Directions and Possible Extensions

In the previous chapters, we have developed a theory for estimating the state of decision and control systems that relies on partial order theory to reduce computational complexity. In Chapter 6, we also showed that such an estimation method allows treating the continuous variables and the discrete variables in the same way as one can construct one partial order that contains both the discrete and the continuous variable spaces. We have proposed a multi-robot system as a guiding example to show how a "good" lattice can be chosen for solving computational complexity issues. However, for an arbitrary system, there is not a general procedure for establishing the partial order that allows reducing the computational burden.

The aim of this section is to show a couple of application examples that are very different from each other and from the multi-robot example, for which the proposed state estimation methodology can be used in order to reach tractability of the estimation problem. In general, distributed and multi-agent systems suffer from the combinatorial explosion of the state space, and state estimation algorithms that scale with the number of agents are often necessary. Thus, we present the following two application examples. The first example is concerned with the state estimation problem in purely discrete event dynamic systems modeled as Petri nets (Section 7.2.1). We show that these systems naturally evolve on a partial order that is interval compatible with the system itself. The second example is a monitoring problem of a distributed environment involving interacting agents whose states

change dynamically, as it happens in constrained human environments. We show how one can choose a coordinate frame for describing the system that is characterized by a partial order with which the transformed system is interval compatible (Section 7.2.2).

7.2.1 State Estimation in Discrete Event Systems Modeled as Petri Nets

A discrete event system is a transition system whose state changes are driven by events. We do not give the details on discrete event system models in this chapter, and the interested reader is referred to [16]. Petri net models are an alternative to automata for representing the dynamics of a discrete event system. They are often used to model manufacturing environments, and they are well suited for representing causal relationships, process synchronization, resource allocation, and concurrency. We define the Petri net model in the following section.

7.2.1.1 Petri Net Model

Like an automaton, a Petri net is a device that manipulates events according to certain rules. One of its features is that it includes explicit conditions under which an event can be enabled; this allows the representation of very general discrete even systems whose evolution depends on potentially complex control schemes. This representation is described graphically, resulting in Petri net graphs. Any automaton can always be represented as a Petri net, but not all Petri nets can be represented as automatons. Consequently, Petri nets represent a larger class of languages than regular languages.

In Petri nets, events are associated with transitions. In order for a transition to occur, several conditions may have to be satisfied. The information about these conditions is contained in *places*. Some such places are viewed as an "input" to a transition, and they contain the information related to the condition for the transition to occur. Other places are viewed as "output" to a transition as they are affected by the transition occurrence. The transitions and places are the basic components of a Petri net graph. In particular, a Petri net graph has two kinds of nodes, places and transitions, and arcs connecting these. It is a

bipartite graph as no two nodes of the same kind can be connected by arcs.

Formally, a Petri net is a weighted bipartite graph (P, T, A, ω, s) (see [9] for details on graph theory), in which $P = \{p_1, ..., p_n\}$ is a set of places, $T = \{t_1, ..., t_m\}$ is a set of transitions, and $A \subseteq P \times T \cup T \times P$ is a set of arcs from places to transitions and from transitions to places. $\omega : A \to \{1, 2, 3...\}$ is a weight function on the arcs, $s : P \to \mathbb{N}$ is the function that assigns to the set of places a state $\mathbf{s} = (s(p_1), ..., s(p_n)) \in \mathbb{N}^n$.

Definition 7.2.1. The input and output places of a transition t_j are denoted $\text{In}(t_j)$ and $\text{Out}(t_j)$, respectively, and are defined as

$$\begin{aligned} \text{In}(t_j) &:= \{p_i \in P \mid (p_i, t_j) \in A\} \\ \text{Out}(t_j) &:= \{p_i \in P \mid (t_j, p_i) \in A\}. \end{aligned}$$

In a similar way, the input and output transitions of a place p_i are denoted $\text{In}(p_i)$ and $\text{Out}(p_i)$, respectively, and are defined as

$$\begin{aligned} \text{In}(p_i) &:= \{t_j \in T \mid (t_j, p_i) \in A\} \\ \text{Out}(p_i) &:= \{t_j \in T \mid (p_i, t_j) \in A\}. \end{aligned}$$

The weight function $\omega : A \to \{1, 2, 3...\}$ is such that if $p_i \notin \text{In}(t_j)$ or $t_j \notin \text{Out}(p_i)$ then $\omega(p_i, t_j) = 0$. If $p_i \notin \text{Out}(t_j)$ or $t_j \notin \text{In}(p_i)$ then $\omega(t_j, p_i) = 0$.

The state transition function $f : \mathbb{N}^n \times \mathcal{P}(T) \to \mathbb{N}^n$ that updates the state $\mathbf{s}$ is defined for transition t_j if and only if t_j is enabled at $\mathbf{s}$. The set of enabled transitions at $\mathbf{s}$ is given by

$$\mathcal{E}(\mathbf{s}) = \{t_j \mid s(p_i) \geq \omega(p_i, t_j) \ \forall \ p_i \in \text{In}(t_j)\}. \tag{7.1}$$

Not all enabled transitions necessarily fire. Then, we denote $\mathcal{F}(\mathbf{s}) \subseteq \mathcal{E}(\mathbf{s})$ to be the set of firing transitions that in fact fire at $\mathbf{s}$. We assume that the set of firing transitions is such that $|\mathcal{F}(\mathbf{s}) \cap \text{Out}(p_i)| \leq 1$. This means that if two transitions are enabled and share the same input place, they cannot fire at the same time. Then, the state transition function f is

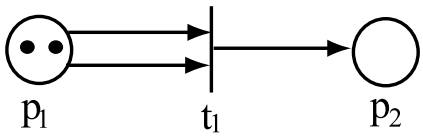


Figure 7.1: In the picture, we have $P = \{p_1, p_2\}$, $T = \{t_1\}$, $A = \{(p_1, t_1), (t_1, p_2)\}$, $\omega(p_1, t_1) = 2$, and $\omega(t_1, p_2) = 1$. Moreover, $\text{In}(t_1) = \{p_1\}$, $\text{Out}(t_1) = \{p_2\}$. The state of the net is given by $\mathbf{s} = (2, 0)$, and in this state, transition t_1 is enabled.

defined according to

$$s'(p_i) = s(p_i) - \sum_{j \mid t_j \in \mathcal{F}(\mathbf{s})} \omega(p_i, t_j) + \sum_{j \mid t_j \in \mathcal{F}(\mathbf{s})} \omega(t_j, p_i). \tag{7.2}$$

An example of a Petri net with two places is represented in Figure 7.1.

Let $\mathbf{s}_k$ denote the state of the net at step k and $\mathcal{F}(\mathbf{s}_k)$ the set of transitions fired at $\mathbf{s}_k$. An execution of the system (P, T, A, ω, s) is the sequence of states $\sigma = \{\mathbf{s}_k\}_{k \in \mathbb{N}}$ with $\mathbf{s}_{k+1} = f(\mathbf{s}_k, \mathcal{F}(\mathbf{s}_k))$. The state lives in $\mathcal{U} = \mathbb{N}^n$. Since we assume to measure the transitions that fire, $\mathcal{F}(\mathbf{s}_k)$, the output sequence is the sequence $y = \{\mathcal{F}(\mathbf{s}_k)\}_{k \in \mathbb{N}}$ where $y_k = \mathcal{F}(\mathbf{s}_k)$. The set in which the output lives is then $\mathcal{Y} \subseteq \mathcal{P}(T)$. The output function g is then defined as $g(\mathbf{s}_k) = \mathcal{F}(\mathbf{s}_k)$. Given a firing sequence enabled at $\mathbf{s}_k$, $\mathcal{F}(\mathbf{s}_k)\mathcal{F}(\mathbf{s}_{k+1})...\mathcal{F}(\mathbf{s}_{k+p})$, we define the notation

$$f(\mathbf{s}_k, \mathcal{F}(\mathbf{s}_k)\mathcal{F}(\mathbf{s}_{k+1})...\mathcal{F}(\mathbf{s}_{k+p})) := f(...f(f(\mathbf{s}_k, \mathcal{F}(\mathbf{s}_k)), \mathcal{F}(\mathbf{s}_{k+1}))..., \mathcal{F}(\mathbf{s}_{k+p})).$$

The set of all possible states compatible with an observed firing is said to be the output set and it is defined in the following definition.

Definition 7.2.2. (Output set) Given the set $y_k \subset T$ of fired transitions at step k, the *output set* at step k is the set of all states $\mathbf{s}$ that enable all of the transitions in y_k, i.e.,

$$O_y(k) := \{\mathbf{s} \in \mathbb{N}^n \mid \mathcal{E}(\mathbf{s}) \supseteq y_k\}.$$

The system is *observable* if whenever two state sequences are different, the correspond-

ing firing sequences are also different. In the following section, we show that the state of a Petri net naturally evolves on a partial order and that the system and such a partial order are interval compatible.

7.2.1.2 State Estimation on a Partial Order

In this section, we introduce the partial order $(\chi, \leq)$ to use for the estimator. The system specified by (7.1) and (7.2) naturally evolves on a partial order, the output set is an interval sublattice on such a partial order, and the dynamics is order isomorphic on the output set. In particular, we establish $\mathcal{U} = \chi = \mathbb{N}^n$ with component-wise order. As a consequence, we have a particular case of the general theory developed in Chapters 3-4, in which $\Sigma = \tilde{\Sigma}$.

The following propositions show that the system $\Sigma = (\mathcal{U}, \mathcal{Y}, f, g)$ specified in the previous section and the partial order $(\chi, \leq)$ are interval compatible. First, we assume that the state of each place is bounded, that is, $s(p_i) \leq u_i$ and $\mathbf{s} \leq u$ for $u = (u_1, ..., u_n)$.

Proposition 7.2.1. *The output set corresponding to the set of firing transitions y_k is an interval sublattice, that is*

$$O_y(k) = [\bigwedge O_y(k), \bigvee O_y(k)]$$

for all k, in which $\bigwedge O_y(k) = l_y(k) = (l_{y,1}(k), ..., l_{y,n}(k))$ and $l_{y,i}(k) = \omega(p_i, t_j)$ for the transition t_j such that $t_j \in y_k \cap Out(p_i)$ and $l_{y,i} = 0$ if $y_k \cap Out(p_i) = \emptyset$. Also, $\bigvee O_y(k) = u$.

Proof. We show that each element of the first set belongs to the second one and *vice versa*. Let us show first that if $\mathbf{s} \in O_y(k)$ then $\mathbf{s} \in [l_y(k), u]$. If $\mathbf{s} \in O_y(k)$, then $y_k \subseteq \mathcal{E}(\mathbf{s})$ by the definition of output set. This, along with the definition of enabled transitions (7.1), implies that for any p_i $s(p_i) \in [\omega(p_i, t_j), u_i]$ for $t_j \in y_k \cap \text{Out}(p_i)$. Also, note that if $y_k \cap \text{Out}(p_i)$ is not empty, there is by assumption only one $t_j \in y_k \cap \text{Out}(p_i)$. If $y_k \cap \text{Out}(p_i) = \emptyset$, it means either that p_i does not enable any transition in $\text{Out}(p_i)$ or that it does but such transition does not fire. As a consequence, if $y_k \cap \text{Out}(p_i) = \emptyset$ it follows that $s(p_i) \in [0, u_i]$. Then, we have showed that if $\mathbf{s} \in O_y(k)$, then $\mathbf{s} \in [l_y(k), u]$.

Assume now that $\mathbf{s} \in [l_y(k), u]$; we want to show that $\mathbf{s} \in O_y(k)$. To show this, it is enough to show that the set of enabled transitions at $\mathbf{s}$ includes all of the transitions y_k, that

is, $\mathcal{E}(\mathbf{s}) \supseteq y_k$. Since $\mathbf{s} \in [l_y(k), u]$, we have that $s(p_i) \geq \omega(p_i, t_j)$ if $t_j \in y_k \cap \text{Out}(p_i)$. As a consequence $t_j \in \mathcal{E}(\mathbf{s})$. This holds for any $t_j \in y_k$ and thus $\mathcal{E}(\mathbf{s}) \supseteq y_k$. □

Proposition 7.2.2. *The state transition function* $f : ([\wedge O_y(k), \vee O_y(k)], y_k) \rightarrow [f(\wedge O_y(k), y_k), f(\vee O_y(k), y_k)]$ *is order isomorphic.*

Proof. By definition of order isomorphic function, and by the way $\wedge O_y(k)$ and $\vee O_y(k)$ have been defined, we have to show that $f : ([l_y(k), u], y_k) \rightarrow [f(l_y(k), y_k), f(u, y_k)]$ is order embedding and onto.

Let us show first that it is order embedding, that is, for any $a, b \in [l_y(k), u]$ we have $a \leq b$, if and only if $f(a, y_k) \leq f(b, y_k)$. Let $a = (a_1, ..., a_n)$ and $b = (b_1, ..., b_n)$. It follows from equations (7.2) that if $a_i \leq b_i$ then $a'_i \leq b'_i$ since the same quantity is added to both a_i and b_i as the set of firing transitions y_k is the same for both of them. Similarly, if $a'_i \leq b'_i$ then $a_i \leq b_i$ for the same reasoning.

Let us show that it is onto. We show that if $a' \in [f(l_y(k), y_k), f(u, y_k)]$ then there is $a \in [l_y(k), u]$ such that $a' = f(a, y_k)$. If $a' \in [f(l_y(k), y_k), f(u, y_k)]$ then

$$l_{y,i}(k) - \sum_{j\,:\,t_j \in y_k} \omega(p_i, t_j) + \sum_{j\,:\,t_j \in y_k} \omega(t_j, p_i) \leq a'_i$$

and

$$a'_i \leq u_i - \sum_{j\,:\,t_j \in y_k} \omega(p_i, t_j) + \sum_{j\,:\,t_j \in y_k} \omega(t_j, p_i).$$

Define a_i such that $a'_i = a_i - \sum_{j\,:\,t_j \in y_k} \omega(p_i, t_j) + \sum_{j\,:\,t_j \in y_k} \omega(t_j, p_i)$, then $l_{y,i}(k) \leq a_i \leq u_i$. □

As a consequence of Propositions 7.2.1 and 7.2.2, the system Σ modeled as a Petri net and the partial order $(\chi, \leq)$ are interval compatible. Note that since $\mathcal{U} = \chi$, we have that $L(k) = \mathbf{s}_k$ and $U(k) = \mathbf{s}_k$ for $k \geq k_0$ for some $k_0 > 0$. This way, the computable quantity $|[U(k) - L(k)]|$ gives a measure of the estimation error, which goes to zero if the system is observable. The opposite statement is also true, that is, if $|[U(k) - L(k)]|$ does not go to zero then the system with initial conditions in $[0, u]$ is not observable. In fact, the set $[L(k), U(k)]$ is by construction the set of all possible states that are consistent with the output observation and with the system dynamics.

If the state of each place is not upper bounded, or such a bound is not known, the previous two propositions transform to the following.

Proposition 7.2.3. *The output set is a $\wedge$-semilattice, that is*

$$O_y(k) = [\bigwedge O_y(k), \infty)$$

where $\wedge O_y(k) = l_y(k) = (l_{y,1}(k), ..., l_{y,n}(k))$, and $l_{y,i}(k) = \omega(p_i, t_j)$ for the transition t_j such that $t_j \in y_k \cap Out(p_i)$ and $l_{y,i} = 0$ if $y_k \cap Out(p_i) = \emptyset$.

Proposition 7.2.4. *The state transition function $f : ([\bigwedge O_y(k), \infty), y_k) \rightarrow [f(\bigwedge O_y(k), y_k), \infty)$ is order isomorphic.*

In the case in which the places are not upper bounded, we have $U(k) = \infty$ in Theorem 3.3.1, and the estimator therein can be constructed with the properties

(i) $\mathbf{s}_k \geq L(k)$ for all k;

(ii) $|[L(k+1), \mathbf{s}_{k+1}]| \leq |[L(k), \mathbf{s}_k]|$ for all k;

(iii') there is $k_0 > 0$ such that for any $k \geq k_0$ we have that $L(k) = \mathbf{s}_k$.

Note that the resulting estimator for this case is the same as the one derived by other means in [30]. This shows that the kinds of estimators for Petri nets as developed in [30] can be naturally re-derived as a particular instance of the partial order based state estimation approach developed in this thesis. The computational complexity of the proposed estimator is proportional to the number of plates as opposed to the dimension of the state space (that is combinatorial in the number of plates). Note that if an upper bound on the state of a place is not known, we do not have a way of computing the estimation error that scales with the number of places as in the case in which such a bound is known.

7.2.2 Monitoring a Distributed Environment

In this section, we consider an example that is very different from the previously proposed ones, for which we show how one can choose a "good" partial order. The purpose

of this section is thus to purely show ways of establishing useful coordinate frames in decision and control systems whose models are fairly general. We do not claim to solve a specific practical problem, which is left for future work. The problem considered here is the one of estimating the state of a multi-agent nondeterministic hybrid system modeling the behavior of agents, for example people, in a common environment such as a building, a hospital, a laboratory, or a manufacturing chain. The measurements come from sensors that, placed at a small number of locations, detect the presence or the absence of a person, without recognizing his or her identity. The environment is partitioned in locations some of which are "interesting" and some of which are not because they are used to move from one location to the other and no interesting activity occurs in them. The problem to solve is the one of establishing at which location each agent is at each time, given the sensor firings and an approximate model of the agent dynamics and decisions. An obvious way to attack this problem is to divide the environment into a grid and at each sensor firing to establish the set of all possible environment configurations (in terms of the grid) compatible with the sensor firings. This leads to combinatorial complexity because the sensors cannot distinguish between agents.

In this section, we show that we can establish a coordinate system that has as each coordinate the position of an agent along its own trajectory. Each agent trajectory can have branching corresponding to possible decisions of the agent. Also, each agent evolves on its own trajectory in a nondeterministic way due to the fact that he can randomly stop, accelerate, or decelerate. Trajectories can be involved in mutual constraints representing meetings between agents. In such a coordinate system, in which the order is established according to the causal order relation ("happened before" relation), the dynamics of the agents preserves the order by construction, and the output set can be approximated by an interval in the partial order.

7.2.2.1 System Model

Consider N agents $\{A_1, ..., A_N\}$ in a common $E \subseteq \mathbb{R}^2$ environment, which is partitioned in a set of *locations* $\mathcal{L} = \{\lambda_0, ..., \lambda_n\}$, with $\lambda_i \subset E$. The locations $\{\lambda_1, ..., \lambda_n\}$ are referred to as "interesting" locations as we are interested in determining what agent is occupying them

(if any). The location λ_0 is everything left of E once the interesting locations have been removed, i.e., $\lambda_0 = E - \{\lambda_1, ..., \lambda_n\}$. Let $p_i \in E$ for each i represent the position of agent i, and let $\alpha_i \in \mathcal{L}$ for each i represent the location of agent i. The state $s_i = (p_i, \alpha_i)$ of agent i is updated according to the laws

$$\begin{aligned} p_i(k+1) &= h_i(p_i(k), \alpha_i(k)) + \Delta h_i(k) \\ \alpha_i(k+1) &= f_i(p_i(k), \alpha_i(k)), \end{aligned} \tag{7.3}$$

subject to

$$P(s(k), k) = true, \qquad s(0) \in S_0, \tag{7.4}$$

in which $h_i : E \times \mathcal{L} \to E$, $f_i : E \times \mathcal{L} \to \mathcal{L}$, $s = (p, \alpha)$ with $\alpha = (\alpha_1, ..., \alpha_N)$ and $p = (p_1, ...p_N)$, Δh_i bounded uncertainties, $P : E^N \times \mathcal{L}^N \times \mathbb{N} \to \{true,\ false\}$ is a predicate that puts global constraints among the agents. If no constraint is imposed, $P = true$ always. The functions (h_i, f_i) are referred to as the nominal dynamics of the agent. The transition functions f_i can be implemented as a set of logic rules (if-then-else). The measurements are given by sensors placed at some of the locations in $\mathcal{L} - \lambda_0$. Each sensor returns 0 if no person is detected, and it returns 1 if a person is detected. Formally, let $\mathcal{L}_s = \{\lambda_{s,1}, ..., \lambda_{s,m}\} \subset \mathcal{L} - \lambda_0$ be the set of locations at which a sensor is positioned, then the measurement is given by $y = g(s)$ taking values in $\mathcal{Y} = \{0, 1\}^m$ such that for each $j \in \{1, ..., m\}$

$$y_j = \begin{cases} 1 & \text{if there is } i \in \{1, .., N\} \text{ with } \alpha_i = \lambda_{s,j} \\ 0 & \text{otherwise.} \end{cases} \tag{7.5}$$

Each agent enters any location through a door that is identified as a point in E. We first assume that there are no uncertainties on the f_i and no measurement errors. We show in a final section how to handle uncertainties and sensor errors within the proposed estimation framework.

Given the output measurement, the knowledge of the nominal dynamics, and bounds on Δh_i, we want to produce an estimate $\hat{s}$ of the system state s that converges to s if there is no uncertainty ($\Delta h_i = 0$ for each i). In case there is uncertainty, we ask that the distance

$d(s, \hat{s})$, for some distance function d, stays bounded possibly at a small value that allows discriminating between interesting locations for each agent. More formally, we have the following estimation problem.

Problem 7.2.1. (Estimation problem) Given the system defined in (7.3), with constraints (7.4), and with output (7.5), determine a function $\phi : E^N \times \mathcal{P}(\mathcal{L}^N) \times \mathcal{Y} \to E^N \times \mathcal{P}(\mathcal{L}^N)$, such that the estimate $\hat{s} = (\hat{p}, \hat{\alpha})$ with $\hat{\alpha} \in \mathcal{P}(\mathcal{L}^N)$ and $\hat{p} \subset E^N$

$$\hat{s}(k+1) = \phi(\hat{s}(k), y(k+1))$$

has the property that there is $k_0 > 0$ such that

(i) $d(p(k), \hat{p}(k)) \leq M$ for any $k \geq k_0$ for some distance function d;

(ii) $\hat{\alpha}(k) \cap (\mathcal{L} - \lambda_0)^N = \alpha(k)$ for any $k \geq k_0$;

and such that if $\Delta h_i = 0$ for each i, we have

(i') $d(p(k), \hat{p}(k)) = 0$ for any $k \geq k_0$;

(ii') $\hat{\alpha}(k) = \alpha(k)$ for any $k \geq k_0$.

In this problem formulation, the estimate of $\alpha(k)$ is the set $\hat{\alpha}(k)$ of all values of the discrete state compatible with the observed output and with the system dynamics. Item (ii) requires that even if there is an estimation error on the continuous variable $p(k)$, such error is small enough to let us disambiguate the interesting locations for each agent. Note also that $\hat{p}(k) \subset E^N$, that is, it is a set of agent positions compatible with the output sequence and with the dynamics.

Note that the state estimation problem has combinatorial complexity in the number of agents as the output firings do not discriminate between agents. In the following section, we reformulate the estimation problem on a lattice to overcome this combinatorial complexity issue. In particular, we transform the system to a lattice where the dynamics preserves the partial ordering and where the output set can be approximated by an interval sublattice.

7.2.2.2 Formulation of the Estimation Problem on a Lattice

In this section, we find a partial order on the system such that the set $\hat{s}$ of all possible system states can be represented or approximated by a lower and an upper bound in the chosen partial order. In particular, we look for a partial order whose ordering is preserved by the system dynamics, the output set can be represented by a interval sublattice, and the constraints (7.4) can be reformulated in terms of lower and upper bounds. In order to do so, we give some preliminary definitions.

Definition 7.2.3. (Abscissa) Let $p : [a, b] \to \mathbb{R}^2$ be the parameterization of a path in $\mathbb{R}^2$. For any $t \in [a, b]$ with $t = a + k$ for some $k \in \mathbb{N}$, the function

$$z(t) = \sum_{\tau=a}^{t-1} \|p(\tau + 1) - p(\tau)\|$$

represents the length of the path covered up to t starting from $p(a)$ increasing t one unit at a time, and it is called *abscissa*.

The function z is a monotonic increasing function of t. If t represents time, all points on such abscissa are ordered according to the causal order relation, i.e., $z(t_1) \leq z(t_2)$ if and only if $t_1 \leq t_2$. Also the inverse function $t(z)$ is monotonic increasing.

In the case of the application under study, each one of the agents that visits a sequence of locations, will have to pass through the doors that are identified with points in E. If $\Delta h_i = 0$, the path covered in the building is fixed, as every agent has to enter the building by a door. As a consequence, each agent has a nominal abscissa intrinsically attached to it, and it will move along it as time goes on. The uncertainty on the initial condition will translate on the uncertainty on where the agent is along the abscissa. The uncertainty Δh_i will result in an uncertainty on how far the agent moves on the abscissa from its current position on the abscissa. Note that in general, depending on the structure of f_i, an abscissa can have branchings corresponding to decisions taken by the agent, with this decision not being directly observed. In order to explain the basic ideas, we initially assume that the abscissa has no branchings. The case in which branchings occur can be treated with minor modifications that are explained in a later section.

Note that when an agent is in one of the interesting locations, we are not interested in tracking exactly its position; as a consequence we will represent as a point on the abscissa each interesting location. This way the not interesting locations λ_0 will appear as connectors between interesting locations (points on the abscissa). Thus, the agent abscissa is formally defined in the following definition.

Definition 7.2.4. (Agent abscissa) Let $p_i : [a, b] \to E$ be the trajectory of agent A_i with $\Delta h_i = 0$ in the environment E with $a \geq 0$ and $t = a + k \leq b$ for some $k \in \mathbb{N}$. The quantity

$$z_i(t) = \sum_{\tau=a}^{t-1} \|\bar{p}_i(\tau + 1) - \bar{p}_i(\tau)\|,$$

with

$$\bar{p}_i(t) = \begin{cases} p_i(t) & \text{if } p_i(t) \in \lambda_0 \\ p(\lambda_j) & \text{if } p_i(t) \in \lambda_j \text{ for } j \neq 0, \end{cases}$$

is the agent A_i abscissa, in which $p(\lambda_j)$ denotes the position of the door of location j. The set of all points on the abscissa is denoted by

$$\gamma_i = \{z_i \mid \exists\, t \in [a, b] \text{ with } t = a + k \text{ for } k \in \mathbb{N} \text{ and } z_i = \sum_{\tau=a}^{t} \|\bar{p}_i(\tau + 1) - \bar{p}_i(\tau)\|\}.$$

In the case in which each location has more than one door, we can still collapse the location at one point. The coordinates along the abscissa of agent A_i corresponding to the location λ_j are denoted by $\lambda_j^i = \{\lambda_j^{i,1}, ..., \lambda_j^{i,n_i}\}$, each associated with a different time the agent visited that location along its trajectory. Also, $\lambda_0^i = \gamma_i - \cup_j \lambda_j^i$. The set of positions along the abscissa at which there is a sensor $j \in \{1, ..., m\}$, is denoted $\lambda_{s,j}^i$.

In these new coordinates, the system state is given by $(z, \alpha) \in \mathcal{Z} \times \mathcal{L}^N$, with $z =$

$(z_1, ..., z_N)$ and $\mathcal{Z} = \gamma_1 \times ... \times \gamma_N$. The new system dynamics is defined for any i as

$$
\begin{aligned}
z_i(k+1) &= z_i(k) + v_i + \Delta v_i(k) && (7.6)\\
\alpha_i(k+1) &= \lambda_j \text{ if } z_i(k+1) \in \lambda^i_j && (7.7)\\
y_j(k+1) &= 1 \text{ if } \exists\ i \text{ with } z_i(k+1) \in \lambda^i_{s,j} && (7.8)\\
y_j(k+1) &= 0 \text{ if } z_i(k+1) \notin \lambda^i_{s,j}\ \forall\ i, && (7.9)\\
z_i(0) &\in [L_{i,0}, U_{i,0}], && (7.10)
\end{aligned}
$$

in which we assume that v_i is known, constant along the abscissa, and any variation to it is incubated in the uncertainty $\Delta v_i(k)$, with $\Delta v_i(k) \in [\Delta v_{i,m}, \Delta v_{i,M}]$. The uncertainty on the initial condition is transformed to an interval not including sensor locations along the abscissa. We do not comment at this point on the constraints specified by (7.4) and devote a later section to them.

This formulation is in accordance with the kind of model that we could have for each persons daily habits. The behavior will typically be described by sentences of the form " agent i enters in the morning between 8am and 10am, then he usually goes to his office, where he has 30 minutes-1 hour meetings scheduled with agents (in the order) j, k, q."

Notice also that, due to the discrete model of the agent motion, the abscissa has a grid of points on which the agent transitions, the abscissa being defined on the basis of the nominal dynamics. In order for the agent with uncertainty $\Delta v_i(k) \neq 0$ to still evolve on the abscissa, we require that $\Delta v_i(k)$ is such that for any k it moves the agent at points on the grid of the abscissa. This technical point is due to our choice of a discrete model for the agent dynamics. This technical point would be abandoned if the equations of motion of the agent were represented in continuous time. We also assume for simplicity that there cannot be two locations next to each other on the abscissa.

In the next definition, we establish a partial order on $\mathcal{Z} \times \mathcal{L}^N$.

Definition 7.2.5. For any pair of elements $w, x \in \mathcal{Z} \times \mathcal{L}^N$ with $w = (z^w, \alpha^w)$ and $x = (z^x, \alpha^x)$

the partial order $(\mathcal{Z} \times \mathcal{L}^N, \leq)$ is established by

$$w \leq x \text{ if and only if } z^w \leq z^x$$

and

$$z^w \leq z^x \text{ if and only if } z_i^w \leq z_i^x \text{ for all } i.$$

Since the partial order is established on the basis of the z component only, and the discrete state α can be unequivocally determined once the continuous one has been found, we continue our arguments considering the z component of the state only. Thus, we establish the partial order $(\chi, \leq)$ with $\chi = \mathcal{Z}$ and partial order as given in Definition 7.2.5. Any $x \leq w \in \mathcal{Z}$ define an interval sublattice in $\mathcal{Z}$ denoted $[x, w] = [x_1, w_1] \times ... \times [x_N, w_N]$. Note that the discrete state evolution established by f_i was used for defining the abscissas γ_i, as f_i establishes the sequence of locations that the agent visits, and h_i establishes how the agent evolves in each location. In the new coordinate system $\mathcal{Z}$, these two different evolutions have been fused together into one variable evolution z_i for each agent. For any $x \leq w \in \mathcal{Z}$, we define the *distance* $d(x, w)$ as follows

$$d(x, w) := \frac{1}{N} \sum_{i=1}^{N} (w_i - x_i), \text{ with } w_i - x_i := |[x_i, w_i] - \{x_i\}|.$$

The system represented by equations (7.6-7.7-7.8-7.9) is denoted $\Sigma = (\mathcal{Z}, \mathcal{Y}, F, g)$, in which F is specified by equation (7.6) and g by equations (7.8-7.9). If $\Delta v_i = 0$ for any i, the dynamics (7.6) preserves the partial order $(\chi, \leq)$, and $F : [L, U] \to [F(L), F(U)]$ is onto. If $\Delta v_i \neq 0$, i.e., the system is nondeterministic, one can verify that $F : [L, U] \to [\wedge F(L), \vee F(U)]$ is order preserving and onto.

Next, we show that the output set can be approximated by an interval in $(\chi, \leq)$. Let again O_y denote the output set corresponding to a measurement y, that is, $O_y = \{z \in \mathcal{Z} \mid g(z) = y\}$. Let $[L, U] \subseteq \mathcal{Z}$ denote an interval sublattice in $\mathcal{Z}$, with $L = (L_1, ..., L_N)$, $U = (U_1, ..., U_M)$, $U_i \in \gamma_i$, and $L_i \in \gamma_i$, such that $|[L_i, U_i] \cap \lambda^i_{s,j}| \in \{1, 0\}$ for each i, j. This means that $[L_i, U_i]$ cannot contain more than one coordinate point on γ_i corresponding to the same location at which a sensor is positioned. Let $O_y|_{[L,U]}$ denote the output set once z has been restricted

to belong to an interval sublattice $[L, U]$, that is, $O_y|_{[L,U]} = \{z \in [L, U] \mid g(z) = y\}$. For simplifying notation, assume that only one sensor can fire at one time. Then, such set has the property shown in the following proposition.

Proposition 7.2.5. *Let y be such that $y_j = 1$ and $y_k = 0$ for any $k \neq j$. Then $\wedge O_y|_{[L,U]} \in O_y|_{[L,U]}$ and $\vee O_y|_{[L,U]} \in O_y|_{[L,U]}$. Let $l_y := \wedge O_y|_{[L,U]}$ and $u_y := \vee O_y|_{[L,U]} \in O_y|_{[L,U]}$, then $l_y = (l_{y,1}, ..., l_{y,N})$ and $u_y = (u_{y,1}, ..., u_{y,N})$ with*

$$l_{y,i} = \begin{cases} \lambda^i_{s,j} \cap [L_i, U_i] & \text{if } \lambda^k_{s,j} \cap [L_k, U_k] = \emptyset \ \forall k \neq i, \\ L_i & \text{if } \lambda^k_{s,j} \cap [L_k, U_k] \neq \emptyset \text{ for some } k \neq i \text{ and } L_i \notin \lambda^i_{s,k} \text{ for } k \neq j, \\ L_i + v_i & \text{if } \lambda^k_{s,j} \cap [L_k, U_k] \neq \emptyset \text{ for some } k \neq i \text{ and } L_i \in \lambda^i_{s,k} \text{ for } k \neq j, \end{cases} \tag{7.11}$$

and

$$u_{y,i} = \begin{cases} \lambda^i_{s,j} \cap [L_i, U_i] & \text{if } \lambda^k_{s,j} \cap [L_k, U_k] = \emptyset \ \forall k \neq i, \\ U_i & \text{if } \lambda^k_{s,j} \cap [L_k, U_k] \neq \emptyset \text{ for some } k \neq i \text{ and } L_i \notin \lambda^i_{s,k} \text{ for } k \neq j, \\ U_i - v_i & \text{if } \lambda^k_{s,j} \cap [L_k, U_k] \neq \emptyset \text{ for some } k \neq i \text{ and } L_i \in \lambda^i_{s,k} \text{ for } k \neq j. \end{cases} \tag{7.12}$$

Proof. This can be easily proved by showing two facts. 1) If $x \in O_y|_{[L,U]}$, then $l_y \leq x$ and $u_y \geq x$. 2) Both L_y and U_y are in $O_y|_{[L,U]}$.

Proof of 1). We proceed component-wise. If $x \in O_y|_{[L,U]}$, then $g(x) = y$ and $x_i \in [L_i, U_i]$. Then, if $l_{y,i} = L_i$ there is nothing to show. If $l_{y,i} = L_i + v_i$, we need to show that there cannot be any $x \in O_y|_{[L,U]}$ with $x_i = L_i$. If $l_{y,i} = L_i + v_i$, then by (7.11), $x_i \in \lambda^i_{s,k}$ for $k \neq j$, that is, it is placed at a sensor location. By the definition of g in equations (7.8-7.9) if $x_i \in \lambda^i_{s,k}$ for some k, it must be $y_k = 1$, which is a contradiction. Finally, consider $l_{y,i} = \lambda^i_{s,j} \cap [L_i, U_i]$ and thus $U_{y,i} = L_{y,i}$. Then, by (7.11) we have that $\lambda^k_{s,j} \cap [L_k, U_k] = \emptyset \ \forall k \neq i$. This, in turn implies that $x_i = \lambda^i_{s,j} \cap [L_i, U_i]$ because agent i is the only one that can have caused that sensor firing.

Proof of 2). This is clear if $l_{y,i} = u_{y,i} = \lambda^i_{s,j} \cap [L_i, U_i]$. If $l_{y,i} = L_i$, we show that there is $x \in O_y|_{[L,U]}$ such that $x_i = L_i$. If there is no x such that $x_i = L_i$ with $g(x) = y$, it means that $L_i \in \lambda^i_{s,k}$ for $k \neq j$, which is a contradiction by the definition (7.11). If $l_{y,i} = L_i + v_i$,

we show that there is $x \in O_y|_{[L,U]}$ such that $x_i = L_i + v_i$. If this is not the case, it means that $L_i + v_i \in \lambda^i_{s,k}$ for $k \neq j$. However, by (7.11) $L_i = \lambda^i_{s,k}$ for $k \neq j$. So we would have both L_i and $L_i + v_i$ corresponding to locations. This contradicts our assumptions that establish that on one abscissa there cannot be locations close to each other. □

This proposition guarantees that the interval sublattice $[l_y, u_y]$ is the smallest interval that contains $O_y|_{[L,U]}$, and thus it is the best representation of such a set in terms of interval sublattices. The reason why $O_y|_{[L,U]} \neq [l_y, u_y]$ is because there are points in $[l_{y,i}, u_{y,i}]$ where agent A_i cannot be. In fact the agent cannot be at coordinate points on the abscissa that correspond to sensor locations at which the senor has not fired. Define for each i the set of points where the sensors have not fired as

$$z_i^{nf} := \cup_j \{\lambda^i_{s,j} \ : \ y_j = 0\}. \tag{7.13}$$

We denote $\mathcal{Z}_{not} := \mathcal{Z} - [(\gamma_1 - z_1^{nf}) \times ... \times (\gamma_N - z_N^{nf})]$. By definition, no state compatible with the output can be in such a set. In case $\Delta v_i = 0$ for each i, such a set changes dynamically in a deterministic way as positions that were not occupied at step k map to positions that cannot be occupied at step $k + 1$. This gives rise to a set $\mathcal{U}(k)$ to which $\mathcal{Z}$ must be constrained. Such a constraint set is defined in the following definition.

Definition 7.2.6. For any execution σ of the system Σ with output sequence $g(\sigma) = \{y(k)\}_{k \in \mathbb{N}}$ and with $\Delta v_i = 0$ for any i, we define the *constraint set* at step k, $\mathcal{U}(k)$, to be the set defined as

$$\mathcal{U}(0) = \mathcal{Z} - \mathcal{Z}_{not}(0),$$

$$\mathcal{U}(k+1) = F(\mathcal{U}(k)) - \mathcal{Z}_{not}(k+1).$$

Note that the notion of constraint set makes no sense if $\Delta v_i \neq 0$ as in such a case we cannot know from one step to another where $\mathcal{U}(k)$ is mapped to.

In case $\Delta v_i \neq 0$, the function F maps a point to a set. The supremum and infimum of this set for any $z \in \mathcal{Z}$ are defined as

$$\bigvee F(z) = z + v + \Delta v_M$$

and

$$\bigwedge\!\!\!\!\bigwedge F(z) = z + v + \Delta v_m.$$

If there is no uncertainty, $\bigvee\!\!\!\!\bigvee F = \bigwedge\!\!\!\!\bigwedge F = F$.

Given the system $\Sigma = (\mathcal{Z}, \mathcal{Y}, F, g)$ with $z(0) \in [L_0, U_0] \subset \mathcal{Z}$ and with output sequence $\{y(k)\}_{k\in\mathbb{N}}$, we can implement the estimator for nondeterministic systems on a partial order presented in Chapter 5, that is,

$$\begin{aligned} L(k+1) &= \bigwedge\!\!\!\!\bigwedge F(L(k)) \curlyvee l_y(k+1) \\ U(k+1) &= \bigvee\!\!\!\!\bigvee F(U(k)) \curlywedge u_y(k+1) \\ L(0) &= L_0,\ U(0) = U_0, \end{aligned} \tag{7.14}$$

in which $l_y(k) = \bigwedge\!\!\!\!\bigwedge O_y(k)|_{[L',U']}$ and $u_y(k) = \bigvee\!\!\!\!\bigvee O_y(k)|_{[L',U']}$ with $L' = \bigwedge\!\!\!\!\bigwedge F(L(k))$ and $U' = \bigvee\!\!\!\!\bigvee F(U(k))$. We also assume that the set $[L(k), U(k)]$ is such that $|[L_i(k), U_i(k)] \cap \lambda^i_{s,j}| \in \{1, 0\}$ for each i, j. This can be verified if L_0 and U_0 are not too far from each other compared to the distances between the coordinate of the sensor locations on the abscissas. If the system is deterministic, that is, $\bigvee\!\!\!\!\bigvee F = \bigwedge\!\!\!\!\bigwedge F = F$, then we have the following result, which is a straightforward consequence of the results in Chapter 3.

Proposition 7.2.6. *Given the system $\Sigma = (\mathcal{Z}, \mathcal{Y}, F, g)$ with output sequence $\{y(k)\}_{k\in\mathbb{N}}$ and with $\Delta v_i = 0$ for any i, then the update laws (7.14) are such that*

(i) $z(k) \in [L(k), U(k)]$ *for any k;*

(ii) $|[L(k+1), U(k+1)]| \leq |[L(k), U(k)]|$*;*

(iii) if Σ is observable, then there is a $k_0 > 0$ such that $[L(k), U(k)] \cap \mathcal{U}(k) = z(k)$ for any $k \geq k_0$.

The only difference with the results in Chapter 3 is that in the present case the set $\mathcal{U}$ is not constant in time. It in fact depends on the measurements while in Chapter 3 it depended on the space structure, which was the case of Chapter 3. Despite this difference, the results follow in the same way.

If the system is nondeterministic, in order to guarantee that $|[L(k), U(k)]|$ is bounded asymptotically, we need to ask for an additional structural property of the system. In fact, observability is not sufficient because the output set is not exactly an interval sublattice as the theory developed in Chapter 5 requires. This additional property, named interval observability, is defined in the following definition.

Definition 7.2.7. (Interval observability) Consider the system $\Sigma = (\mathcal{Z}, \mathcal{Y}, F, g)$ with output sequence $\{y(k)\}_{k\in\mathbb{N}}$, and consider the update laws as in equations (7.14). The system Σ is said to be *interval observable* if for any i there is an infinite sequence $\{k_{i,0}, ..., k_{i,l}, ...\}$ such that

(i) $U_i(k_{i,l}) \in \lambda^i_{s,j}$ for some j;

(ii) $[L_i(k), U_i(k)] \cap \lambda^i_{s,j} \neq \emptyset \Rightarrow [L_p(k), U_p(k)] \cap \lambda^p_{s,j} = \emptyset$ for any $p \neq i$, for any $k_{i,l} \leq k \leq k_{i,l} + \Delta k_{i,l}$, with $k_{i,l} + \Delta k_{i,l}$ such that $L(k_{i,l} + \Delta k_{i,l}) \in \lambda^i_{s,j}$.

This property basically requires that periodically the set $[L_i(k), U_i(k)]$, for any i, will be the only one to contain the coordinate corresponding to the sensor j for all the interval of time that $[L_i(k), U_i(k)]$ contains such a coordinate. This guarantees that the set $[L(k), U(k)]$ does not grow unbounded due to nondeterminism as the following proposition shows.

Proposition 7.2.7. *Given the system $\Sigma = (\mathcal{Z}, \mathcal{Y}, F, g)$ with output sequence $\{y(k)\}_{k\in\mathbb{N}}$, then the update laws (7.14) are such that*

(i) $z(k) \in [L(k), U(k)]$ for any k;

(ii) if Σ is interval observable, then there is a $k_0 > 0$ such that $d(L(k), U(k)) \leq M$ with

$$M = \frac{1}{N}\sum_{i=1}^{N} \min\left(d_i \frac{\Delta v_{i,M} - \Delta v_{i,m}}{v_i + \Delta v_{i,m}}, d_i\right),$$

where $d_i = \max_l(U(k_{i,l+1}) - U(k_{i,l}))$.

Proof. For each i, let $d_i := max_j(U(k_{i,j+1}) - U(k_{i,j}))$. Then, given the update laws (7.14), we have that agent A_i takes between $d_i/(v_i + \Delta v_{i,M})$ and $d_i/(v_i + \Delta v_{i,m})$ steps to cover the

distance d_i, then we have that

$$U_i(k) - L_i(k) \leq \min\left(d_i \frac{\Delta v_{i,M} - \Delta v_{i,m}}{v_i + \Delta v_{i,m}}, d_i\right).$$

□

Let $d_{i,m}$ be the minimum distance between interesting locations on the abscissa γ_i. Problem 3.2.1 is solved if the uncertainties on the sets $[L_i, U_i]$ are such that

$$\min\left(d_i \frac{\Delta v_{i,M} - \Delta v_{i,m}}{v_i + \Delta v_{i,m}}, d_i\right) \leq d_{i,m} \ \forall \ i. \tag{7.15}$$

Note that there are two ways to satisfy such inequality:

(1) Act on the sensor position such that d_i is decreased. For example, one can put sensors such that between the points on the abscissa corresponding to $U(k_{i,j})$ and $U(k_{i,j+1})$, there is only one interesting location.

(2) The previous measure is not needed if the smallest velocity of the agent $v_i + \Delta v_{i,m}$ is high compared to the distance d_i. In this case, the agents excite the sensor with high frequency and thus the uncertainty is decreased (we show this point in a simulation example).

7.2.2.3 Meeting Constraints on the Partial Order

We consider two kinds of meeting constraints that we call of type C1 and of type C2.

C1. Two or more agents meet in one location specifically dedicated to the meeting, that is, none of the agents can be at such location alone. Formally, let $A_{i_1}, ..., A_{i_P}$ be the agents involved in such meeting constraint, then the constraint P can be formulated as

$$\forall \ p \in \{1, ..., P\}, \ \alpha_{i_p}(k) = \lambda_j \text{ if and only if } \alpha_{i_q}(k) = \lambda_j \text{ for any } q \neq p. \tag{7.16}$$

C2. One or more agents go to meet some other agent to a location already occupied by the latter agent. For example students meet a professor in his office. This means that the students cannot be in the professor's office unless the professor is there. Formally, let A_q be the agent in location λ_j and $\{A_{i_1}, ..., A_{i_P}\}$ be different agents meeting A_q in λ_j. Then, the constraint P can be set as

$$\forall\ p \in \{1, ..., p\}\ \alpha_{i_p}(k) = \lambda_j\ \Rightarrow\ \alpha_q(k) = \lambda_j. \tag{7.17}$$

In the partial order formulation on $(\mathcal{Z}, \leq)$ these two types of constraint simply translate to conditions on lower and upper bounds according to what follows.

C1. Let $A_{i_p} \in [L_{i_p}, U_{i_p}]$ and let $\lambda_j^{i_p}$ any abscissa coordinate corresponding to the location l_j, then (7.16) reduces to

$$U_{i_p} \leq \lambda_j^{i_p} \quad \Leftrightarrow \quad U_{i_q} \leq \lambda_j^{i_q}\ \forall\ p, q \in \{1, ..., P\}, \tag{7.18}$$

$$L_{i_p} \geq \lambda_j^{i_p} \quad \Leftrightarrow \quad U_{i_q} \geq \lambda_j^{i_q}\ \forall\ p, q \in \{1, ..., P\}, \tag{7.19}$$

C2. and (7.17) reduces to

$$U_q \leq \lambda_j^q \quad \Rightarrow \quad U_{i_p} \leq \lambda_j^{i_p}\ \forall\ p, q \in \{1, ..., P\}, \tag{7.20}$$

$$L_q \geq \lambda_j^q \quad \Rightarrow \quad L_{i_p} \geq \lambda_j^{i_p}\ \forall\ p, q \in \{1, ..., P\}. \tag{7.21}$$

In the next section, we give a qualitative overview on how to deal with uncertainty issues.

7.2.2.4 Dealing with Uncertainty on the Model, Random Disturbances, and Measurement Errors

In this section, we consider uncertainty originating from different sources: uncertainty on the model of the discrete state evolution (uncertainty on f_i), random disturbances caused for example by unmodeled agents that populate the environment, and measurement errors

due to false alarms or missed detections. The arguments are not formal and have the sole purpose of showing possible ways of dealing with uncertainty within this framework.

Uncertainty on the Model. In this case, given a current agent location, (1) either the next location is not uniquely determined and belongs to a set of possible known locations, (2) or only a nominal next location is known and the rest is unmodeled because it is unexpected.

In case (1), the abscissa of an agent looks as depicted in Figure 7.2. This could correspond in practice to the fact that the agent does not behave exactly the same way every day, but there might be variations from one day to the other that one can model. Also, the condition that decides which way to go is not directly observable in general. In this case nothing changes in the estimation algorithm structure except that for each agent, the algorithm has to keep track of all possible branchings at the same time. This translates to a lower and an upper bound with dynamic dimension for each agent. The dimension increases when a new branching occurs, and it decreases when one branching becomes inconsistent with the measurement. The drawback is that the algorithm updates more than one variable for each agent with an increased computational cost.

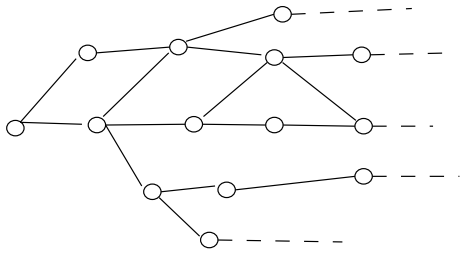


Figure 7.2: Abscissa of one agent with uncertainty on the model of type (1).

In case (2), the abscissa of each agent looks as depicted in Figure 7.3. The unshaped parts are completely unknown as they correspond to the agent behavior that was not expected and thus not modeled. If an agent goes in the unshaped region, it is lost until (if ever) he comes back on the nominal path. This corresponds in practice to the fact that one day an agent does not enter the building because he is sick or something unexpected happened to him, for example. In this case, the agent is lost. However, *robustness of the*

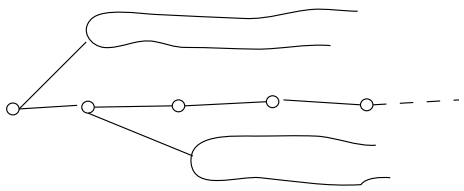


Figure 7.3: Abscissa of one agent with uncertainty on the model of type (2).

estimator requires that this does not affect "too much" the estimation error on the other agents. This can be guaranteed under the assumption already made of interval observability. This assumption states that for each agent i there is periodically one sensor firing for which the interval $[L_i, U_i]$ is the only one compatible with the abscissa coordinate where the sensor firing occurs for the entire time $[L_i, U_i]$ is compatible with it. As a consequence, if a firing does not occur for any time at which $[L_i, U_i]$ is the only one compatible with the firing of the sensor, it means that agent A_i did not follow the nominal path. This fact gives an idea of how the algorithm can detect when an agent does something unexpected. Once the inconsistency is detected for one agent, the algorithm can keep estimating the position of all the other agents as usual. Note that before the inconsistency is detected, the estimation error can increase for all of the agents (this point is illustrated in a later simulation example).

Random Disturbances. This paragraph covers the case in which there are people wandering around the building whose identities are not known and whose behaviors are not modeled. They obviously cause the sensors to fire when they stop by locations at which the sensors are positioned. *Robustness of the estimator requires that the estimation error does not diverge due to these random events*. This can be obtained for example if an agent returns periodically to the same location where a sensor is placed after a long enough period of time. This way, if a firing is caused by a random agent, this happening will be detected at some later time. The basic idea is that when a firing occurs and a random agent could be the cause of that, the algorithm can keep track of multiple hypotheses, one in which the firing was caused by an agent and one in which the firing was caused by a random agent. This causes an increase of the estimation error. At some later time the false hypothesis

will be detected as such, and the estimation error decreases again (this is illustrated in a simulation example).

Measurement Errors. Measurement errors can be of two kinds. We can have missed detections or false alarms. The case of missed detection can be treated in a way similar to the way uncertainty on the model (case (2)) is treated. In fact, a missed detection will cause an inconsistency to be detected as a sensor firing was in fact expected, but it did not occur. The case of false alarm is similar to the case of random firings caused by random agents. These "spurious firings" can be detected as such as explained in the paragraph on random disturbances.

7.2.2.5 Simulation Examples

We conclude the example of environment monitoring with some simulation examples, which give an idea of how the performance of the proposed algorithm looks like. We give some simulation results for both the deterministic case ($\Delta v_i = 0$) and the nondeterministic case ($\Delta v_i \neq 0$ without uncertainty on the model of f_i). We then, show how the algorithm copes with the case in which we have uncertainty on the model of type (2), i.e., an agent never entered the building, and with the case in which there are random agents that make sensors fire. We assumed that initially all of the agents were outside of the building in an interval along their abscissas between L_0 and U_0 (corresponding to uncertainty on when the agent usually enters the building). The entrance of the building has a sensor that detects a person passing through it. Also, we distributed sensors along the abscissas. In the figures, we show the error

$$E(k) = \frac{1}{N}\sum_{i=1}^{N} E_i(k), \text{ with } E_i(k) = (U_i(k) - L_i(k)).$$

In Figure 7.4, we consider an example without uncertainty for different numbers of agents. As the number of agents increases, the time taken to convergence increases as the number of sensor firings needed for disambiguating the agents increases as well. In Figure 7.5, we show the error $E(k)$ in the nondeterministic case for two different values of $\Delta v_{m,i} + v_i$.

When such a value is increased, the upper bound on the error decreases as predicted by our analysis (see Proposition 7.2.7).

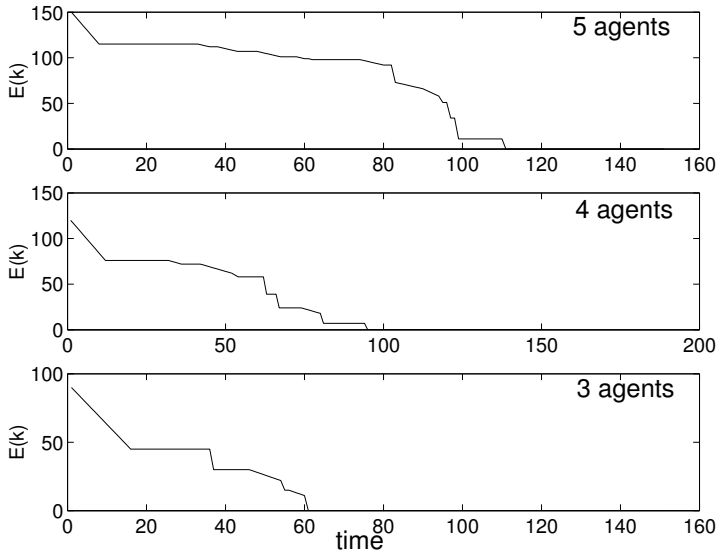


Figure 7.4: Example without dynamic uncertainty: convergence plots for different numbers of agents.

In Figure 7.6, we show the case in which an agent never entered the building. The algorithm detects an inconsistency as explained in the paragraph on uncertainty on the model. In Figure 7.7, we show the case in which a random agent makes the sensors fire and the algorithm detects it as explained in the paragraph on random disturbances.

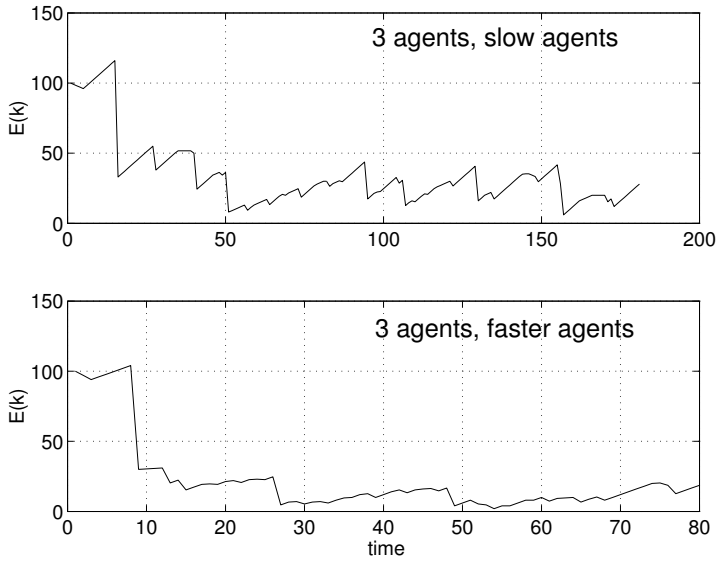


Figure 7.5: Example with dynamic uncertainty: convergence plots for different values of $\Delta v_{i,m} + v_i$. The lower plot has a value three times the one of the upper plot for each agent. For any agent, the nominal speed is $v_i = 1$ and $\Delta v_{i,m} = 0$ and $\Delta v_{i,M} = 2$, so that the speed of each agent is uniformly distributed in $[1, 3]$.

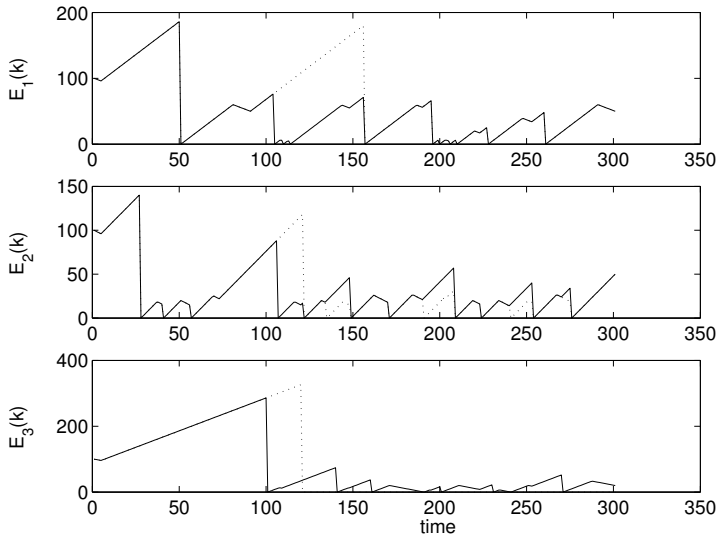


Figure 7.6: Example in which agent 3 never enters the building. The solid line shows the error we would have if agent 3 entered the building as expected. The dashed line shows the error in the case that agent 3 unexpectedly does not enter the building. The error on agents one and two grows with respect to the nominal case until the inconsistency is detected (at k=120). At this point the lower and upper bounds for agent three are arbitrarily set to zero indicating that the agent has been lost.

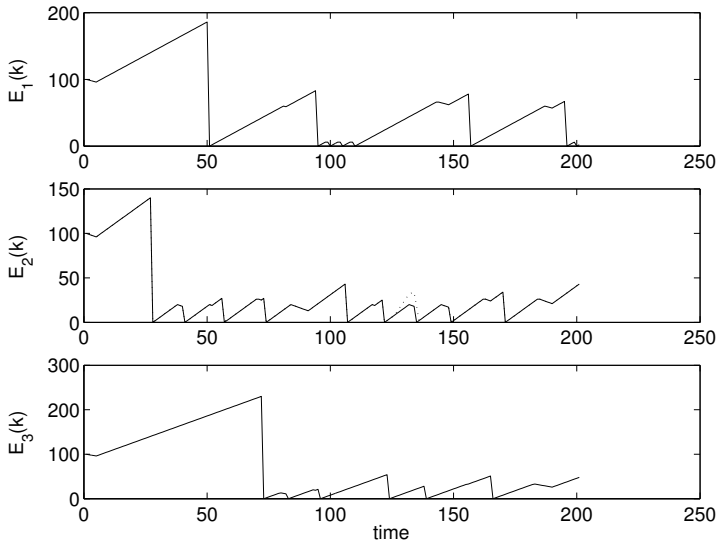


Figure 7.7: Example with a random agent causing firing of sensors between $k = 100$ and $k = 150$. The dashed line shows the error with such random firings. The solid line shows the error we would have without the random firings. Note that after the inconsistency is detected, the error goes back to normal.

Bibliography

[1] S. Abramsky and A. Jung. *Domain Theory*. Handbook of Logic in Computer Science Volume 3. 1994.

[2] A. Alessandri and P. Coletta. Design of luenberger observer for a class of hybrid linear systems. In *Lecture Notes in Computer Science* 2034, M. D. Di Benedetto, A. Sangiovanni-Vincentelli Eds. Springer Verlag, pages 7–18, 2001.

[3] A. Alessandri and P. Coletta. Design of observer for switched discrete-time linear systems. In *American Control Conference*, pages 2785–2790, 2003.

[4] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. Sangiovanni-Vincentelli. Design of observers for hybrid systems. In *Lecture Notes in Computer Science* 2289, C. J. Tomlin and M. R. Greenstreet, Eds. Springer Verlag, pages 76–89, 2002.

[5] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. Sangiovanni-Vincentelli. Observability of hybrid systems. In *Conf. on Decision and Control*, pages 1159–1164, 2003.

[6] A. Bayen, J. Zhang, C. Tomlin, and Y. Ye. Milp formulation and polynomial time algorithm for an aircraft scheduling problem. In *American Control Conference*, 2003.

[7] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control*, 45:1864–1876, 1999.

[8] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. SIAM, 1994.

[9] B. Bollobas. *Modern Graph Theory*. Springer, 1998.

[10] P. Brucker, M. Garey, and D. Johnson. Treelike precedence constraints to minimize maximum lateness. *Math. Oper. Res.*, pages 275–284, 1977.

[11] H. H. Bui, S. Venkatesh, and G. West. Policy recognition in the abstract hidden markov model. *Journal of Artificial Intelligence Research*, 17:451–499, 2002.

[12] P. E. Caiens and Y. Wei. The hierarchical lattices of a finite machine. *Systems and Control Letters*, 25:257–263, 1996.

[13] P. E. Caines. Classical and logic-based dynamic observers for finite automata. *IMA J. of Mathematical Control and Information*, pages 45–80, 1991.

[14] P. E. Caines and S. Wang. Cocolog: A conditional observer and controller logic for finite machines. *SIAM J. Control and Optimization*, pages 1687–1715, 1995.

[15] A. R. Cassandra, L. P. Kaelbling, and M. L Littman. Acting optimally in partially observable stochastic domains. In *Proc. 12th Conference on Artificial Intelligence*, pages 1023–1028, Seattle, WA, 1994.

[16] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Klwer Academic Publisher, 1999.

[17] R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10):1456–1477, 2001.

[18] E. F. Costa and J. B. R. do Val. On the observability and detectability of continuous-time jump linear systems. *SIAM Journal on Control and Optimization*, 41:1295–1314, 2002.

[19] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, 1977.

[20] R. D'Andrea, R. M. Murray, J. A. Adams, A. T. Hayes, M. Campbell, and A. Chaudry. The RoboFlag Game. In *American Control Conference*, pages 661–666, 2003.

[21] B. A. Davey and H. A. Priesteley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[22] C. M. Özveren and A. S. Willsky. Observability of discrete event dynamic systems. *IEEE Transactions on Automatic Control*, 35(7):797–806, 1990.

[23] D. DelVecchio and R. M. Murray. Discrete state estimators for a class of hybrid systems on a lattice. In *Lecture Notes in Computer Science* 2993, R. Alur and G. J. Pappas Eds. Springer Verlag, pages 311–325, 2004.

[24] D. DelVecchio and R. M. Murray. Discrete state estimators for a class of nondeterministic hybrid systems on a lattice. In *Conf. on Decision and Control*, pages 3215 – 3220, 2004.

[25] D. DelVecchio and R. M. Murray. Existence of cascade discrete-continuous state estimators for systems on a partial order. In *Lecture Notes in Computer Science* 3414, M.Morari and L. Thiele Eds. Springer Verlag, pages 311–325, 2004.

[26] D. DelVecchio and R. M. Murray. Existence of discrete state estimators for hybrid systems on a lattice. In *Conf. on Decision and Control*, pages 1– 6, 2004.

[27] D. DelVecchio, R. M. Murray, and P. Perona. Decomposition of human motion into dynamics based primitives with application to drawing tasks. *Automatica*, 39:2085–2098, 2003.

[28] E. DeSantis, M. D. Di Benedetto, and G. Pola. On observability and detectability of continuous-time linear switching systems. In *Conf. on Decision and Control*, pages 5777–5782, 2003.

[29] M. Diaz, G. Juanole, and J. Courtiat. Observer-a concept for formal on-line validation of distributed systems. *IEEE Trans. on Software Engineering*, 20:900–913, 1994.

[30] A. Giua and C. Seazu. Observability of place/transition nets. *IEEE Transactions on Automatic Control*, pages 1424–1437, 2002.

[31] P. Godefroid. *Partial-order methods for the verification of concurrent systems*. Lecture notes in computer science. Springer, 1996.

[32] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2001.

[33] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[34] E. Klavins. A formal model of a multi-robot control and communication task. In *Conf. on Decision and Control*, pages 4133–4139, Hawaii, 2003.

[35] E. Klavins and R. M. Murray. Distributed algorithms for cooperative control. *Pervasive Computing*, 3:56–65, 2004.

[36] D. G. Luenberger. An introduction to observers. *IEEE Transactions on Automatic Control*, AC-16:6:596–602, 1971.

[37] M. Oishi, I. Hwang, and C. Tomlin. Immediate observability of discrete event systems with application to user-interface design. In *Conf. on Decision and Control*, pages 2665 – 2672, Hawaii, 2003.

[38] P. J. Ramadge. Observability of discrete event systems. In *Conf. on Decision and Control*, pages 1108–1112, Athens, Greece, 1986.

[39] K. Rudie, S. Lafortune, and F. Ling. Minimal communication in a distributed discrete event system. *IEEE Trans. on Automatic Control*, 48:957–975, 2003.

[40] H. L. Smith. *Monotone Dynamical Systems*. Mathematical Surveys and Monographs. American Mathematical Society, 1995.

[41] E. D. Sontag. *Mathematical Control Theory*. Springer, 1998.

[42] C. Tomlin, I. Mitchell, and R. Ghosh. Safety verification of conflict resolution maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 2(2):110–120, 2001.

[43] R. Vidal, A. Chiuso, and S. Soatto. Observability and identifiability of jump linear systems. In *Conf. on Decision and Control*, pages 3614 – 3619, Las Vegas, 2002.

[44] R. Vidal, A. Chiuso, S. Soatto, and S. Sastry. Observability of linear hybrid systems. In *Lecture Notes in Computer Science* 2623, O. Maler and A. Pnueli Eds. Springer Verlag, pages 526–539, 2003.

[45] Y. Zeng, W. Cai, S. J. Turner, S. Zhou, and B. Lee. Characterization and delivery of directly coupled causal messages in distributed systems. *Future Generation Computer Systems*, pages 171 – 178, 2004.