Discrete event system methods for control problems arising in cyber-physical systems

by

Eric Dallal

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2014

Doctoral Committee:

Professor Stéphane Lafortune, Chair
Associate Professor Domitilla Del Vecchio, MIT
Professor Jessy W. Grizzle
Assistant Professor Gabor Orosz
Assistant Professor Necmiye Ozay
Professor Demosthenis Teneketzis

ACKNOWLEDGEMENTS

I would like to thank my advisor Stéphane Lafortune, my colleagues in publications Stéphane Lafortune, Alessandro Colombo and Domitilla Del Vecchio, my committee members Stéphane Lafortune, Domitilla Del Vecchio, Jessy Grizzle, Gabor Orosz, Necmiye Ozay, and Demosthenis Teneketzis, my colleagues Yi-Chin Wu and Xiang Yin, my parents Ilan and Linda, my sister Caroline, my friends, and anyone/anything else I may have bounced ideas off of, including my pet tortoise, Koopa.

The work contained in this thesis was supported in part by NSF grant CNS-0930081, NSF grant CCF-1138860 (Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering) and by a fellowship from Fonds FQRNT, Government of Québec, Canada. The work on MPOs also benefited from useful discussions with Franck Cassez, Stavros Tripakis and Tae-Sic Yoo.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

LIST OF FIGURES vi

LIST OF TABLES x

LIST OF ABBREVIATIONS xi

ABSTRACT xii

CHAPTER

I. Introduction 1

1.1 Discrete Event Systems Preliminaries 4
1.2 Problem Descriptions 11
1.2.1 Dynamic Diagnosability 11
1.2.2 Vehicle Control 13
1.2.3 Problem Comparison 16
1.3 Related Literature 17
1.3.1 Dynamic Diagnosability 17
1.3.2 Vehicle Control 18
1.4 Solution Methodology & Contributions 21
1.4.1 Dynamic Diagnosability 21
1.4.2 Vehicle Control 24

II. Dynamic Diagnosability 33

2.1 Abstract 33
2.2 Introduction 34
2.3 Problem Formulation 37
2.4 Towards an Information State 41
2.5 The State Disambiguation Problem and the Most Permissive Observer 46

2.6 The Extended Specification and Properties of the MPO . . . 54
2.7 Reducing the Information State . . . 59
2.8 Constructing the MPO . . . 66
2.8.1 Experimental Results . . . 69
2.9 Conclusion . . . 71

III. Vehicle Control : The case of perfect measurement . . . 83

3.1 Abstract . . . 83
3.2 Introduction . . . 84
3.3 Model and Problem Definition . . . 88
3.4 Bad Set Description . . . 91
3.5 Discrete Abstraction . . . 92
3.6 State Reductions and Supervisory Control . . . 95
3.6.1 Preliminaries . . . 95
3.6.2 The State Reduction . . . 96
3.6.3 The Exact State Reduction . . . 102
3.7 Supervisor Computation and Relations Between the Time-discretized and Discrete Event Systems . . . 105
3.7.1 Supervisory Control Theory of DES . . . 105
3.7.2 Translating Between Transition Systems and Discrete Event Systems . . . 106
3.7.3 Relations Between the Time-discretized and Discrete Event Systems . . . 107
3.8 Algorithmic Implementation . . . 111
3.9 Simulation Results . . . 115
3.9.1 Simulation Descriptions . . . 115
3.9.2 Results & Analysis . . . 118
3.10 Conclusion . . . 120

IV. Vehicle Control : The case of imperfect measurement . . . 135

4.1 Abstract . . . 135
4.2 Introduction . . . 136
4.3 Model . . . 140
4.4 Results from the Case of Perfect Measurement . . . 144
4.4.1 The Initial Abstraction . . . 145
4.4.2 State Reductions & Exact State Reductions . . . 146
4.5 Observers and State Estimation . . . 150
4.5.1 The modified abstraction . . . 151
4.5.2 State Estimate Reductions & Exact State Estimate Reductions . . . 153
4.6 Conditions for State Estimate Reductions and Exact State Estimate Reductions . . . 166

4.6.1 Translating Between Transition Systems and Discrete Event Systems 167
4.6.2 Proofs of state estimate reductions between the observer and the continuous estimator 172
4.7 Conclusion 176

V. Conclusion & Future Work 182

5.1 Dynamic Diagnosability 182
5.2 Vehicle Control 184

BIBLIOGRAPHY 186

LIST OF FIGURES

Figure

1.1 In the above examples, f is a fault event and all other events are observable. Left: A system that is 1-Diagnosable. Right: A system that 2-Diagnosable but not 1-Diagnosable. 13

1.2 A summary of the approach to the construction of the Most Permissive Observer (MPO). The K-Diagnosability problem is mapped to the state disambiguation problem, for which the extended specification is computed. An appropriate information state is defined and the Total Observer (TO), containing all admissible controllers, is constructed over the space of information states. By proving a monotonicity property over the extended specification, we are able to reduce the information state. Finally, the MPO is constructed over the space of reduced information states, obtained as a sub-automaton of the TO, and using the extended specification to determine the safety of control actions and reduced information states. 21

1.3 A depiction of the solution method in the case of perfect measurement. The continuous system is discretized in time and space and a Discrete Event System (DES) abstraction G is defined over the discrete state space. The problem specifications are translated to a sub-automaton H, and a supervisor for the abstracted system is obtained by solving problem Basic Supervisory Control Problem in the Non-Blocking case (BSCP-NB). Finally, the continuous domain supervisor σ is obtained from the supervisor S of the DES domain. 24

1.4 A depiction of the real time system operation of the vehicle control system. The state $x(t)$ is sampled at times $0, \tau, 2\tau, \ldots$, yielding $x(k\tau)$. This sample is then discretized through the function $\ell(\cdot)$ to a lattice point in the set $\tilde{Q}$. The discrete state $\ell(x(k\tau))$ is sent to the supervisor, which allows a subset of the available control actions. One of these control actions is chosen by the vehicles, and held for the following interval of length τ (i.e., a zero-order hold). Finally, the chosen control action v_c, in addition to the actions of the uncontrolled vehicles v_{uc} and the effect of the disturbance d determine the system trajectory. 26

1.5 The solution method. Given the time discretized system of Eq. (1.1) with the space discretization given by $\ell(\cdot)$, constituting system S_b, we can construct DES G that is a state reduction of this system. Given system S_b with measurement uncertainty given by $L(\cdot)$, constituting partially observed system S_b', we can construct DES G' that models the measurement uncertainty of $L(\cdot)$ by partitioning the set of measurements X into equivalence classes Λ. Given the estimator $\overline{S}_b$ of partially observed system S_b', we can construct DES $\overline{G}$ that is a state estimate reduction of $\overline{S}_b$. Furthermore, when $e_{\max} = k\mu\tau/2$ for some $k \in \mathbb{N}$, $\overline{G}$ can be obtained as the observer of G'. A DES supervisor S is then computed by solving problem BSCP-NB, from which a continuous domain supervisor σ is obtained. 28
1.6 A depiction of the process by which the modified DES abstraction G' is constructed from DES G. The set of continuous measurements X is partitioned into a set of equivalence classes with respect to the function $\ell(L(\cdot))$, yielding the set of equivalence classes Λ representing measurement events. The events of U_c are classified as controllable and observable, the events of U_{uc} and W are classified as uncontrollable and unobservable, and the events of Λ are classified as uncontrollable but observable. With four classes of events, the language of G' becomes $\mathcal{L}(G') \subseteq (\Lambda U_c U_{uc} W)^*$. 31
2.1 A finite state automaton with fault event f. 43
2.2 The augmented automaton for the automaton of Fig. 2.1 43
2.3 The total observer for the automaton of Fig. 2.1, with events classified as follows: $E_o = \emptyset$, $E_s = \{a, b\}$, $E_{uo} = \{t\}$, and $E_f = \{f\}$. . . . 53
2.4 The MPO for the automaton of Fig. 2.1, for any $K \geq 1$. We used the convention of *Cassez and Tripakis* (2008) by marking Y states with squares and Z states with circles. 53
2.5 A finite state automaton. Events are classified as follows: $E_o = \emptyset$, $E_s = \{a, b\}$, $E_{uo} = \{t\}$, and $E_f = \{f\}$. 64
2.6 The MPO corresponding to the automaton of Fig. 2.5, with $K = 2$, not using reduced information states. 64
2.7 The MPO corresponding to the automaton of Fig. 2.5, with $K = 2$, using reduced information states. 65
3.1 An example of the vehicle control problem. 84
3.2 An example scenario involving three vehicles on five roads. Blue lines are drawn for each vehicle indicating starting road and ending road. 91
3.3 The transition function ψ. 94
3.4 A system and its corresponding state reduction. States of the left system with the same output are placed in a common box. We use the usual DES convention of denoting marked states with a double circle and initial states with an incoming arrow that has no source state. 102

3.5 A depiction of the state reduction (left) and exact state reduction (right) for a simple system $S_b = (\{1,\ldots,8\}, \{u\}, \rightarrow_b, \{A, B\}, H_b)$, where $H_b(x) = A$ for $x \in \{1,\ldots,4\}$ and $H_b(x) = B$ for $x \in \{5,\ldots,8\}$. In both the left and right cases, there is a transition $(x, u, x') \in \rightarrow_b$ with $x \in H_b^{-1}(A)$ and $x' \in H_b^{-1}(B)$, and hence a transition from A to B in the corresponding state reduction. The system on the right contains some transition $(x, u, x') \in \rightarrow_b$ with $x \in H_b^{-1}(A)$, *for every* $x' \in H_b^{-1}(B)$. For the system on the left, the occurrence of a transition from A to B in the state reduction allows us to determine that S_b is in state 7. For the system on the right, this transition only allows to determine that the system is some state in the set $H_b^{-1}(B)$. 104

3.6 The intersection and vehicle paths used in each of the simulations of this section. Blue lines are drawn for each vehicle indicating starting road and ending road. 116

3.7 The capture sets of Eqs. (3.64)-(3.73) in the open (left) and closed (right) cases. The blue square denotes the bad set. The set of Eqs. (3.64)-(3.67) is depicted with solid lines, and its inflation by $\mu\tau/2$ is depicted in dashed lines. Right: If d_{min} and d_{max} are integer multiples of μ, then Eqs. (3.72) and (3.73) are unnecessary, which is shown by the dotted lines. 134

4.1 The system process in real time. At times $0, \tau, 2\tau, \ldots$, a measurement $\chi \in X$ is obtained. This measurement is used to correct the previous state estimate I^{pred} through $I = I^c(I^{pred}, \chi) = I^{pred} \cap L(\chi)$. The corrected state estimate I is then discretized to a lattice with spacing $\tau\mu$ as $\ell(I)$, and $\ell(I)$ is sent to the supervisor. The controlled vehicles then choose some control action v_c allowed by the supervisor, and this control action, along with the state estimate I and knowledge of the *bounds* on the actions of the disturbance and the uncontrolled vehicles (shown with dashed lines because they are not observed) are used to predict a new state estimate of possible positions at the next time instant that is a multiple of τ, given by $I^{pred} = I^p(I, v_c)$, where I^p is the function of Eq. (4.5). 144

4.2 The transition function ψ of DES automaton G, consisting of three layers, one for each of the event categories U_c, U_{uc}, W, and separated by intermediate states. 146

4.3 An example system S_b. Rectangles are used to denote states with the same output. 166

4.4 The system S_a that is an exact state reduction of system S_b of Fig. 4.3. 166

4.5 The solution method. Given the time discretized system of Eq. (4.2) with the space discretization given by $\ell(\cdot)$ of Eq. (4.3), constituting system S_b, we can construct DES abstraction G that is a state reduction of this system. Given system S_b with measurement uncertainty given by $L(\cdot)$ of Eq. (4.4), constituting partially observed system S'_b, we can construct DES abstraction G' that models the measurement uncertainty of $L(\cdot)$ by partitioning the set of measurements X into equivalence classes Λ, in accordance with Proc. IV.27. Given the estimator $\overline{S}_b$ of partially observed system S'_b, we can construct DES abstraction $\overline{G}$ that is a state estimate reduction of $\overline{S}_b$. Furthermore, when $e_{\max} = k\mu\tau/2$ for some $k \in \mathbb{N}$, $\overline{G}$ can be obtained as the observer of G' (Props. IV.31 and IV.34). A DES supervisor S is then computed by solving problem BSCP-NB, from which a continuous domain supervisor σ is obtained which solves Prob. IV.1 (Thms. IV.32 and IV.35). 177

LIST OF TABLES

Table

2.1 Simulation results showing running times and MPO sizes with and without using the reduced information states, for randomly generated automata with 75 and 100 states. 70
3.1 Scenario 1: No uncontrolled vehicles, no disturbance. 119
3.2 Scenario 2: Uncontrolled vehicles, no disturbance. 120
3.3 Scenario 3: Disturbance, no uncontrolled vehicles. 120

LIST OF ABBREVIATIONS

DES Discrete Event System

CPS Cyber-Physical System

MPO Most Permissive Observer

TO Total Observer

BSCP-NB Basic Supervisory Control Problem in the Non-Blocking case

LTL Linear Temporal Logic

CTL Computational Tree Logic

ABSTRACT

Discrete event system methods for control problems arising in cyber-physical systems

by

Eric Dallal

Chair: Stéphane Lafortune

This thesis considers two problems in cyber-physical systems. The first is that of dynamic fault diagnosis. Specifically, it is assumed that a plant model is available in the form of a discrete event system containing special fault events whose occurrence we would like to diagnose. Furthermore, it is assumed that there exist sensors that can be turned on or off (for example, to save energy) and capable of detecting some subset of the system's non-faulty events. The problem to be solved consists of constructing a compact structure, called the most permissive observer (MPO), containing the set of all sequences of sensor activations that ensure that any fault event's occurrence will be correctly diagnosed within some finite number k of event occurrences. We solve this problem by defining an appropriate notion of information state summarizing the information obtained from the past sequence of observations and sensor activations. The resulting MPO has a better space complexity than that of the previous approach in the literature.

The second problem considered in this thesis is that of controlling vehicles through an intersection. Specifically, we wish to obtain a supervisor for the vehicles that is safe

(i.e., collision-free), non-deadlocking (i.e., ensures that all vehicles eventually cross the intersection, never reaching a state where the supervisor allows no control actions) and maximally permissive (i.e., allows any control action that does not violate safety or non-deadlockingness). Furthermore, we solve this problem in the presence of uncontrolled vehicles, bounded disturbances in the dynamics, and measurement uncertainty. Our approach consists of discretizing the system in time and space, obtaining a discrete event system (DES) abstraction, solving for maximally permissive supervisors in the abstracted domain, and refining the supervisor to one for the original, continuous, problem domain. We provide general results under which this approach yields maximally permissive memoryless supervisors for the original system and show that, under certain conditions, the resulting supervisor will be maximally permissive over the class of all supervisors, not merely memoryless ones. Our contributions are as follows. First, by constructing DES abstractions from continuous systems, we can leverage the supervisory control theory of DES, which is well-suited to finding maximally permissive supervisors under safety and non-blocking constraints. Second, we define a number of relations between transition systems and their abstractions: state reduction, exact state reduction, state estimate reduction, and exact state estimate reduction. These are general notions which allow for the characterization of obtained supervisors as maximally permissive among the class of memoryless supervisors, or maximally permissive among all supervisors.

CHAPTER I

Introduction

This Ph.D. thesis considers two problems in Cyber-Physical System (CPS)s, namely that of dynamic sensor activation for diagnosis, and that of vehicle control at an intersection. Before describing these two problems, we briefly describe what a CPS is. From a high-level perspective, a CPS is a collection of cooperating computational elements that control physical components. Such systems are typically characterized by the distributed nature of their physical components, which are tightly linked to the computational elements. These systems may have centralized or decentralized control architectures, where the choice of one over the other results in trade-offs involving algorithm complexity, processing speed, communication system complexity, etc. Examples of CPSs include wireless sensor networks, autonomous automotive systems, and distributed robotic systems.

The first problem under consideration in this thesis is that of dynamic sensor activation for diagnosis. Briefly, it is assumed that we possess an accurate model of possible system behavior, including that of fault occurrences, along with a limited number of sensors that can provide information about the system's transitions. The goal in this work is to find strategies that allow for the timely diagnosis of any fault occurrence, while minimizing sensor utilization. Minimizing sensor utilization may be desirable for reasons of energy (if the sensors operate with limited access to a power

source), bandwidth (if the sensors readings must be transmitted to other devices), or security (if the sensor readings may reveal sensitive tactical information, such as in unmanned aerial vehicles).

The second problem under consideration is that of supervising vehicles at an intersection. This problem is examined in the presence of uncontrolled vehicles (vehicles whose behavior cannot be restricted or in any way altered by the control system), disturbances representing unmodeled dynamics, and measurement error. The goal is to design a supervisor that ensures that the vehicles fully cross the intersection without collisions. The supervisor must also be minimally restrictive, in the sense that it only interferes with driver behavior when necessary. The last decade has seen a number of commercially available systems designed to assist drivers. Examples of these include: collision detection systems to either warn the driver or stop the car when an object is spotted in the vehicle's path; parking assist systems; and lane change assist systems that warn drivers of vehicles in or entering their blind spot.

Our solutions to both problems are developed using the theory of Discrete Event Systems (DES), and specifically of supervisory control. A DES is a system whose dynamics (i.e., its state transitions) are determined by event occurrences. An example of a discrete event system is a text editor software program. In this example, inputs are obtained from the mouse or the keyboard and generate responses from the software. Thus, a mouse click or a key stroke would constitute an *event* in this example. Notably, the state of the program does not change in between these events, and the time that passes between consecutive events doesn't affect its final state. Thus we can say that such a system is event-driven, rather than time-driven.

DES theory uses elements also found in computer science, such as languages and automata, and adds a control element in the form of supervisory control theory. Specifically, supervisory control theory is the problem of restricting the set of behaviors of a system (given by a language) to some subset of those behaviors, given that

some events may be uncontrollable (i.e., cannot be prevented), unobservable (i.e., are not seen by the system), or both. In the context of this thesis, the dynamic diagnosis problem is formulated directly within the DES formalism. The vehicle control problem, on the other hand, is formulated within the context of classical control of systems represented by differential equations. Our solution to the vehicle control problem translates the problem to one of supervisory control through a technique called *abstraction*.

This Ph.D. thesis is divided into five chapters. Chapters II, III, and IV describe, respectively, our work on dynamic diagnosability, our work on the vehicle control problem in the case of *perfect* measurement, and our work on the vehicle control problem in the case of *imperfect* measurement. Chapter I is this introduction and is divided into three sections. Because both problems being worked on are in the domain of DES, we begin by presenting preliminaries that are necessary to the understanding of this research (Sec. 1.1). We proceed by giving a high-level description of both research problems that make up this work (Sec. 1.2). Following this, we provide an overview of the related literature for both problems under consideration (Sec. 1.3). Finally, we provide an overview of our solution methodology and contributions (Sec. 1.4). It should be noted that there will be overlap between the contents of this introduction and that of Chapters II, III, and IV, particularly in the introduction of the respective chapters. Chapters I is meant to provide a high-level overview of the problems worked on and our solutions and do not contain the same level of detail as Chapters II, III, and IV. The results of Chapter II have appeared in *Dallal and Lafortune* (2014). Earlier versions of the results presented in Chapters III and IV have appeared in *Dallal et al.* (2013a) (in the case of perfect measurement) and *Dallal et al.* (2013b) (in the case of imperfect measurement).

1.1 Discrete Event Systems Preliminaries

This section describes those concepts in discrete events systems (DESs) which are necessary to understand the work in chapters II, III, and IV. It should not be seen as an overview or an introduction to the field. Only matter that is relevant to the two problems that make up this work will be presented here. The emphasis will be on logical automata as a DES modelling formalism, the problem of diagnosability, and the problem of control. See chapters 1-3 of *Cassandras and Lafortune* (2008) for a reference of the following material.

Informally, a DES is an event-driven, discrete space, system in which changes in state occur upon the occurrence of an *event*. Among the modelling formalisms for DESs are automata and Petri nets. This work uses only logical automata, which are sufficient when event occurrences are not synchronized to a clock (as in timed automata), do not have preconditions (sometimes called guards, as in hybrid systems) and do not follow any probabilistic structure (as in stochastic systems). Following is a formal definition of deterministic logical automata.

Definition I.1 (Deterministic Logical Automata)**.** A deterministic logical automaton G is a tuple $G = (X, E, f, x_0, X_m)$, where X is the set of discrete states, E is the set of events, $f : X \times E \to X$ is a *partial* transition function, x_0 is the initial state of the system, and X_m is a set of marked states. For any $x \in X$ and $e \in E$, $f(x, e)$ signifies that the system moves to state $f(x, e)$ when event e occurs in state x. The set of marked states X_m generally represent the set of states where some operation of interest has completed. This set is sometimes omitted from the definition of an automaton if it is not required. □

Henceforth, we will refer to deterministic logical automata only as automata.

Given a set of events E (also called an alphabet), a *string* s is a sequence of zero or more events. The string of length zero is called the *empty string*, and is denoted

by ε. Given two strings s and t (either of which could be empty), the notation st or $s.t$ denotes their concatenation. In this case, s is a *prefix* of st, and t is a *suffix* of st. A *language* L is a set of strings defined over an alphabet E. The $(\cdot)^*$ operation, called the *Kleene-closure* is defined by $L^* := \{\{\varepsilon\} \cup L \cup LL \cup \ldots\}$, where $LL = \{uv : u, v \in L\}$. Thus, the set E^* contains all strings consisting of members of the alphabet E. For any $L \subseteq E^*$, the set $\overline{L}$, called the *prefix-closure*, contains the set of all prefixes of all strings in L and is defined by $\overline{L} := \{s \in E^* : st \in L\}$. Since t could be the empty string in the previous definition, it follows that $L \subseteq \overline{L}$. If $L = \overline{L}$, we call L *prefix-closed*.

The transition function f of an automaton can be extended to strings (rather than only events) by defining $f(x, \varepsilon) := x$ and $f(x, se) := f(f(x, s), e)$, for any string $s \in E^*$. This allows us to define the language and marked language of an automaton G as:

$$\begin{aligned} \mathcal{L}(G) &:= \{s \in E^* : f(x_0, s) \text{ is defined}\} \quad \text{(language of } G\text{)} \\ \mathcal{L}_m(G) &:= \{s \in \mathcal{L}(G) : f(x_0, s) \in X_m\} \quad \text{(marked language of } G\text{)} \end{aligned}$$

From the above definitions, it follows that $\overline{\mathcal{L}(G)} = \mathcal{L}(G)$ and that $\overline{\mathcal{L}_m(G)} \subseteq \mathcal{L}(G)$. An automaton is *blocking* if $\overline{\mathcal{L}_m(G)} \subset \mathcal{L}(G)$. Informally, this means there exist strings in G that cannot be extended to form a marked string in G. Mathematically, $\exists s \in \mathcal{L}(G) : (\nexists t \in E^*)$ s.t. $st \in \mathcal{L}_m(G)$. conversely, an automaton is non-blocking if $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$.

The remainder of this section will deal with problems in DES related to either partial observability (i.e., in the presence of events whose occurrence is not observable), or partial controllability (i.e., in the presence of events whose occurrence is not controllable).

Given some set of events E, there may be some events in E that cannot be

observed. In this case, the set E is partitioned as $E = E_o \cup E_{uo}$, $E_o \cap E_{uo} = \emptyset$, where E_o is the set of events that are observable and E_{uo} is the set of events that are unobservable. In this case, the occurrence of some string $s \in \mathcal{L}(G)$ will result in the observance of the string $P(s)$, called the *projection* of s, where the function $P : E^* \to E_o^*$ is defined by:

$$\begin{aligned} P(\varepsilon) &:= \varepsilon \\ P(e) &:= \begin{cases} e & \text{if } e \in E_o \\ \varepsilon & \text{if } e \notin E_o \end{cases} \\ P(se) &:= P(s)P(e) \end{aligned}$$

Intuitively, the projection P acts as a filter, leaving only those events that are in E_o. We can also define the inverse projection $P^{-1} : E_o^* \to 2^{E^*}$ by $P^{-1}(t) = \{s \in E^* : P(s) = t\}$.

Given some automaton $G = (X, E, f, x_0)$ (i.e., without a set of marked states defined), and a partition of E into $E = E_o \cup E_{uo}$, $E_o \cap E_{uo} = \emptyset$, we may wish to determine when some significant event $e_f \in E_{uo}$ has occurred. This is called the diagnosability problem, or the fault diagnosability problem (if e_f represents the occurrence of a faulty or abnormal event). Let L_f be defined as the set of strings in the language of G in which the event e_f has occurred. That is, $L_f := \{s \in \mathcal{L}(G) : s = s_1 e_f s_2\}$. Given some observed string $t \in E_o^*$, we can determine that e_f has occurred if $P^{-1}(t) \cap \mathcal{L}(G) \subseteq L_f$ and we can determine that e_f has not occurred if $P^{-1}(t) \cap L_f = \emptyset$. If neither condition is satisfied, then we cannot make any determination. The standard diagnosability problem requires only that we "eventually" be able to make the determination.

Definition I.2 (Diagnosability)**.** Given automaton $G = (X, E, f, x_0)$ with observable event set $E_o \subseteq E$, the unobservable event $e_f \in E_{uo} = E \setminus E_o$ is not diagnosable with

respect to $\mathcal{L}(G)$ if it is possible to find strings $s_Y, s_N \in \mathcal{L}(G)$ satisfying the following three conditions:

1. s_Y contains e_f and s_N does not
2. s_Y is of arbitrarily long length after e_f
3. $P(s_Y) = P(s_N)$.

□

Given automaton $G = (X, E, f, x_0)$, the problem of finding a minimal set $E_o \subseteq E$ of observable events such that unobservable event e_f is diagnosable with respect to $\mathcal{L}(G)$ is called the static sensor selection problem for diagnosability. The first problem that this thesis will address is the dynamic sensor selection problem for K-diagnosability (the meanings of dynamic and K-diagnosability will be explained in Sec. 1.2.1).

Next, we describe the supervisory control problem in DES. Once again, we begin with an automaton $G = (X, E, f, x_0, X_m)$ and a partition on the set of events $E = E_c \cup E_{uc}$ with $E_c \cap E_{uc} = \emptyset$. This time, however, E_c and E_{uc} represent controlled and uncontrolled events. A controlled event can be disabled to prevent undesired behavior; an uncontrolled event can never be disabled. A *supervisor* in DES is a function $S : \mathcal{L}(G) \to 2^E$ which chooses which events to enable (i.e., allow to occur) after each string in $\mathcal{L}(G)$. *The supervisor only chooses which events can occur, not which event will occur.* We call S/G the automaton G under the control of S. The language generated by S/G is denoted by $\mathcal{L}(S/G)$ and defined as follows:

1. $\varepsilon \in \mathcal{L}(S/G)$
2. $[(s \in \mathcal{L}(S/G)) \text{ and } (s\sigma \in \mathcal{L}(G)) \text{ and } \sigma \in S(s)] \Leftrightarrow [s\sigma \in \mathcal{L}(S/G)]$

The language marked by S/G is defined by $\mathcal{L}_m(S/G) = \mathcal{L}(S/G) \cap \mathcal{L}_m(G)$. The supervisory control problem in DES consists of finding a supervisor S such that

$\mathcal{L}(S/G) \subseteq L_a \subseteq \mathcal{L}(G)$ and/or $\mathcal{L}_m(S/G) \subseteq L_{am} \subseteq \mathcal{L}_m(G)$, where L_a and/or L_{am} is called the *specification* of the system. In the case of L_a, it represents the system's legal behavior in the sense that S must disable any string outside of L_a. L_{am} may similarly be interpreted as the legal *marked* behavior. Next, we define what it means for a language to be controllable.

Definition I.3 (Controllability)**.** Given a prefix closed language $M = \overline{M}$ over event set E and a subset of uncontrollable events $E_{uc} \subseteq E$, language K is called *controllable* with respect to M and E_{uc} if:

$$\overline{K}E_{uc} \cap M \subseteq \overline{K}.$$

□

In words, the above definition means that if some string $s \in \overline{K}$ (s is legal) can be extended by uncontrollable event $e \in E_{uc}$ (e cannot be disabled), and string $se \in M$ (se is possible), then we should also have $se \in \overline{K}$ (se is legal). If we take $M = \mathcal{L}(G)$ and $K = L_a$ then we obtain that L_a is controllable with respect to $\mathcal{L}(G)$ and E_{uc} if the extension of any string $s \in L_a$ by an uncontrollable event $e \in E_{uc}$ that is also feasible ($se \in \mathcal{L}(G)$) results in a string that is also in L_a ($se \in L_a$). The controllability property is used in the following theorem.

Theorem I.4 (Controllability Theorem)**.** *Given automaton $G = (X, E, f, x_0, X_m)$, uncontrollable event set $E_{uc} \subseteq E$, and specification $K \subseteq \mathcal{L}(G)$, there exists a supervisor S such that $\mathcal{L}(S/G) = \overline{K}$ if and only if K satisfies the controllability condition.*

If the controllability condition is not satisfied, then we instead seek a supervisor S that achieves as much of $\overline{K}$ as possible without allowing any strings outside of $\overline{K}$. Specifically, we seek to find a language $K_{\text{con}} \subseteq \overline{K}$ satisfying the following two properties:

1. $\overline{K_{\text{con}}}E_{uc} \cap \mathcal{L}(G) \subseteq \overline{K_{\text{con}}}$ (i.e., K_{con} is controllable).

2. Given any $K'_{\text{con}} \subseteq \overline{K}$ satisfying $\overline{K'_{\text{con}}}E_{uc} \cap \mathcal{L}(G) \subseteq \overline{K'_{\text{con}}}$, we have the inclusion $\overline{K_{\text{con}}} \supseteq \overline{K'_{\text{con}}}$.

In words, we would like to find a sublanguage of $\overline{K}$ that is controllable and supremal, in the sense that it contains all other controllable sublanguages of $\overline{K}$. Since set inclusion does not induce a total order over sublanguages of $\overline{K}$, it is not obvious that a solution will always exist. It can be shown, however, that a solution to the problem does indeed always exist. The key to proving this is showing that controllability is preserved under union. This solution is called the *supremal controllable sublanguage* of K with respect to uncontrollable event set E_{uc} and language $\mathcal{L}(G)$ and is denoted by $K^{\uparrow C}$. This language appears in the solution to the basic supervisory control problem (BSCP), presented below:

Problem I.5 (BSCP: Basic Supervisory Control Problem)**.** Given automaton $G = (X, E, f, x_0, X_m)$, uncontrollable event set $E_{uc} \subseteq E$, and specification $L_a = \overline{L_a} \subseteq \mathcal{L}(G)$, find a supervisor S such that:

1. $\mathcal{L}(S/G) \subseteq L_a$

2. Given any other supervisor S' satisfying $\mathcal{L}(S'/G) \subseteq L_a$, we have the inclusion $\mathcal{L}(S/G) \supseteq \mathcal{L}(S'/G)$.

□

Condition 2 above is often referred to as *maximal permissiveness*. The solution to problem BSCP is to choose S such that $\mathcal{L}(S/G) = L_a^{\uparrow C}$.

If we would like the supervisor S to be non-blocking (i.e., $\mathcal{L}(S/G) = \overline{\mathcal{L}_m(S/G)}$), then we must use the following theorem instead.

Theorem I.6 (Non-Blocking Controllability Theorem)**.** *Given automaton* $G = (X, E, f, x_0, X_m)$*, uncontrollable event set* $E_{uc} \subseteq E$*, and specification* $K \subseteq \mathcal{L}(G)$*, there exists a non-blocking supervisor* S *such that* $\mathcal{L}_m(S/G) = K$ *and* $\mathcal{L}(S/G) = \overline{K}$ *if and only if* K *satisfies the two conditions:*

1. *Controllability:* $\overline{K}E_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$
2. $\mathcal{L}_m(G)$*-closure:* $K = \overline{K} \cap \mathcal{L}_m(G)$.

Suppose that there exists some automaton H such that $\mathcal{L}_m(H) = K$ and $\mathcal{L}(H) = \overline{K}$. If we interpret $\overline{K}$ as the legal specification and K as the legal marked specification, then the $\mathcal{L}_m(G)$-closure condition can be seen as one of consistency. It means that any string s in both $\overline{K}$ (legal specification) and $\mathcal{L}(G)$ (feasible behavior) should either be marked in both $\mathcal{L}(H)$ and $\mathcal{L}(G)$ or not marked in either.

As before, if we are given a non-controllable legal marked specification $L_{am} \subseteq \mathcal{L}_m(G)$ then we can instead try to find a maximally permissive non-blocking supervisor:

Problem I.7 (BSCP-NB: Basic Supervisory Control Problem - Nonblocking Case)**.** Given automaton $G = (X, E, f, x_0, X_m)$, uncontrollable event set $E_{uc} \subseteq E$, and $\mathcal{L}_m(G)$-closed specification $L_{am} \subseteq \mathcal{L}_m(G)$, find a non-blocking supervisor S such that:

1. $\mathcal{L}_m(S/G) \subseteq L_{am}$
2. Given any other non-blocking supervisor S' satisfying $\mathcal{L}_m(S'/G) \subseteq L_{am}$, we have the inclusion $\mathcal{L}_m(S/G) \supseteq \mathcal{L}_m(S'/G)$.

□

The solution to problem BSCP-NB is to choose S such that $\mathcal{L}(S/G) = \overline{L_{am}^{\uparrow C}}$ and $\mathcal{L}_m(S/G) = L_{am}^{\uparrow C}$.

1.2 Problem Descriptions

This section describes the two problems under consideration in this thesis. Section 1.2.1 describes the dynamic diagnosability problem in discrete event systems and Sec. 1.2.2 describes the vehicular control problem. Section 1.2.3 briefly outlines similarities and differences between the two problems.

1.2.1 Dynamic Diagnosability

Recall the definition of diagnosability from Def. I.2. As stated in section 1.1, the static sensor selection problem in this context consists of finding a minimal set of sensors $E_o \subseteq E$ such that some fault event $e_f \in E$ is diagnosable in the language of some automaton G. This problem can be extended to the dynamic case in which sensors may be activated and deactivated after each observation by the system. In this work, we are concerned with characterizing the entire set of solutions to the problem. Informally, a control decision at any time consists of a set of sensors to monitor. Once a control decision is made, the system enters a waiting state, "waking up" from this state upon the occurrence of an event that it has chosen to monitor, referred to as an *observation*. A new control decision may be made at this point, after which the system enters a waiting state again, and so on. If we denote the sequence of control decisions by $C_0, C_1, \ldots$ and the sequence of observations by $e_0, e_1, \ldots$ then, placing these in chronological order, we obtain an alternating sequence $C_0, e_0, C_1, e_1, \ldots$. A sequence of this form with n control decisions and n observations is called a history, or *run* of length n. In the most general sense, a dynamic controller is a function $C : R \to 2^E$, where R is the set of runs, E is the set of events, and $C(\rho) = \gamma$ signifies that the controller chooses to monitor (i.e., activates sensors for) the set of events $\gamma \subseteq E$ after run $\rho \in R$. A controller is called *safe* if it maintains the K-diagnosability property throughout system operation. Informally, this means the controller must diagnose the occurrence of any fault within $K + 1$ events after a fault. A formal

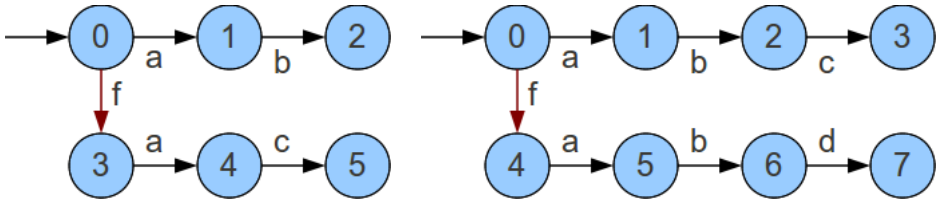
definition is given below, after defining the *controller induced projection*.

For a static set of observable events, the string that is observed upon the occurrence of execution s is the projection $P(s)$. In the dynamic sensor activation problem, the set of observable events changes through the system's execution in a way that depends on controller C. Therefore, we use the notation $P_C(s)$ rather than $P(s)$ to denote the observed string, in order to emphasize the dependence on the controller C. Given some string of events s and controller C, we can iterate through the events of s until we find an event $e_0 \in C_0 = C(\varepsilon)$, where ε denotes the empty history. The event e_0 will be the first event observed in the string s. The next set of events to monitor will be $C_1 = C((C_0, e_0))$. We can then continue to iterate through the events of s (from where we left off), until we find event $e_1 \in C_1$, and so on. See Def. II.3 for a formal definition of the controller induced projection $P_C(\cdot)$. The K-diagnosability property is formally defined below. Also see Fig. 1.1 for an example.

Definition I.8 (Dynamic K-Diagnosability)**.** Given automaton $G = (X, E, f, x_0)$ and controller $C : R \to 2^E$ which chooses which events to monitor after each possible run $\rho = (C_0, e_0, \ldots, C_{n-1}, e_{n-1})$, the unobservable event $e_f \in E$ is not diagnosable with respect to $\mathcal{L}(G)$ if it is possible to find strings $s_Y, s_N \in \mathcal{L}(G)$ satisfying the following three conditions:

1. s_Y contains e_f and s_N does not
2. s_Y is of length at least $K + 1$ after e_f
3. $P_C(s_Y) = P_C(s_N)$

For a fixed automaton $G = (X, E, f, x_0)$ and fault event $e_f \in E$, controller C is called *safe* if it satisfies the K-diagnosability property through any execution of the system (i.e., if there do not exist s_N and s_Y satisfying the three conditions of Def. I.8 for controller C). Individual control decisions are called safe if they allow



(a) A 1-Diagnosable System (b) A 2-Diagnosable System

Figure 1.1: In the above examples, f is a fault event and all other events are observable. Left: A system that is 1-Diagnosable. Right: A system that 2-Diagnosable but not 1-Diagnosable.

for the K-diagnosability property to be maintained through some future sequence of (run-dependent) control decisions.

This work has three goals:

1. To characterize the set of all safe controllers in some finite-sized structure
2. To efficiently compute the set of safe control decisions after any given run
3. To minimize the size of the structure in the first goal and the running times of algorithms for computing safe controllers and safe control decisions.

1.2.2 Vehicle Control

The vehicle control problem consists of finding controllers to safely coordinate a number vehicles across an intersection. Vehicles drive on multiple roads, all leading to a central intersection. We assume that vehicles maintain a strictly positive velocity at all times. Given a set of vehicles $\mathcal{N} = \{1, \ldots, n\}$ and a set of roads $\mathcal{R} = \{1, \ldots, m\}$, vehicle $i \in \mathcal{N}$ enters the intersection on road $r_{i,1}$ and exits the intersection on road $r_{i,2}$. For any road $r \in \mathcal{R}$, the portion of the road that is inside the intersection has size α_r. If we denote vehicle i's position by x_i then vehicle i enters the intersection on road $r_{i,1}$ when $x_i = -\alpha_{r_{i,1}}$, switches instantaneously from road $r_{i,1}$ to road $r_{i,2}$ when $x_i = 0$, and exits the intersection when $x_i = \alpha_{r_{i,2}}$. See Fig. 3.2 for a depiction of the vehicle control problem. The safety criteria are defined as follows:

1. Vehicles on the same road must maintain a minimal distance of at least γ at all times.

2. Vehicles on different roads cannot simultaneously be in the intersection.

Note that the first safety criterion implies that passing of one vehicle by another on the road is not allowed. The set of all vehicle positions that do not satisfy both safety criteria is called the *bad set*, denoted by B.

If we take $x \in X$ to be the vector of vehicle positions and v to be the applied control, then the dynamics of the system are given by the first order model $\dot{x} = v + d$, where d is a disturbance with known bounds representing unmodelled dynamics. That is, $d \in D = [d_{min}, d_{max}]^n$ at all times. It is assumed that the set V is discretized by some parameter $\mu \in \mathbb{R}_+$. That is, each $v \in V$ is a vector whose components are each members of some finite set $\{\mu a, \mu(a+1), \ldots \mu b\}$, for some $a, b \in \mathbb{N}$.

Additionally, we assume that a subset of the vehicles are uncontrolled. Their available control actions are the same as for the other vehicles, but their behaviour is inherently uncontrollable. We do not make any assumptions on the behaviour of these vehicles. As such, it is not even possible to affect their behaviour indirectly through those vehicles that are controllable. Thus, collisions involving two uncontrolled vehicles do not count as a failure of the controller to enforce safety. In chapter III, we assume that vehicle positions are measured perfectly. In chapter IV, we assume that vehicle positions are measured with a maximal error of $e_{\max} \in \mathbb{R}_+$.

As in the dynamic diagnosability problem, we seek the set of all solutions rather than one particular solution. Thus, we wish to obtain a *supervisor* $\sigma : X \to 2^{V_c}$ ($\sigma : 2^X \to 2^{V_c}$ in the case of imperfect measurement) which determines a set of velocities the controlled vehicles are *allowed* to take rather than which precise velocity they will take. The supervisor σ should satisfy three conditions:

1. Safety: If the controlled vehicles choose control actions allowed by σ at each state x, the system avoids the bad set, no matter what happens with the uncontrolled vehicles or the disturbance

2. Non-blockingness: If $\sigma(x(0))$ is non-empty, then the vehicles must eventually cross the intersection, never reaching any state x where $\sigma(x) = \emptyset$

3. Maximal Permissiveness: The supervisor allows any action that can not eventually cause a violation of safety or non-blockingness.

Ideally, we would be able to solve the problem as defined. In practice, the domain over which the supervisor is defined is the continuous set X, which is uncountably infinite. Thus, instead of trying to find a maximally permissive solution over an uncountably infinite domain, we reduce the domain by a suitable discretization in space and time, to obtain a DES. This means restricting the space of possible supervisors σ. Since the maximally permissive criterion for σ is defined relative to the domain of possible supervisors, restricting the space of possible supervisors to some set Σ will mean that the obtained solution σ will not be maximally permissive in the original, continuous-time, continuous space domain.

This work has three goals:

1. To construct a suitable DES abstraction of the original system

2. To translate solution requirements and specifications from the continuous level to the DES level, solve at the DES level, and translate back

3. To characterize the class of supervisors over which the obtained supervisor is maximally permissive.

1.2.3 Problem Comparison

Both of the problems worked on in this thesis are discrete event system control problems: the dynamic diagnosability problem is formulated in discrete event systems terminology whereas the vehicle control problem is solved by discretization in space and time, resulting in a discrete event system abstraction (see Sec. 1.4.2). The two problems are also similar in the sense that a maximally permissive solution is sought to both. In the vehicle control problem, a maximally permissive solution is sought so as to restrict the behavior of the vehicles as little as possible. In the dynamic diagnosability problem, a maximally permissive solution can be used as a first step to obtaining an optimal solution (according to some cost criterion) or simply a minimal solution in the sense of set inclusion (i.e., a Pareto optimal solution). Finally, both problems can be formulated as supervisory control problems in discrete event systems. In the dynamic diagnosability problem, the maximally permissive solution (the most permissive observer) can be obtained as the automaton that marks the supremal controllable sublanguage of the total observer (a structure that contains all possible controllers, including those which are not safe) with respect to a particular specification. In practice, however, the problem will be solved through a different method that allows for faster computation. In the vehicle control problem, formulation as a supervisory control problem is the chosen solution method (see Sec. 1.4.2). Much of the work there focuses on translating safety requirements to the discrete event level, on obtaining efficient algorithms for computing the supremal controllable sublanguage, and on defining the class of supervisors in which the obtained supervisor is maximally permissive.

1.3 Related Literature

1.3.1 Dynamic Diagnosability

The original definition of diagnosability is from *Sampath et al.* (1995). Related literature in the context of *static* sensor selection problems in discrete event systems include e.g., *Haji-Valizadeh and Loparo* (1996), *Jiang et al.* (2003) and *Yoo and Lafortune* (2002). The sensor selection problem is of particularly great importance in wireless sensor networks due to energy limitations. For a survey of work in this domain, see *Rowaihy et al.* (2007). See *Zaytoon and Sayed Mouchaweh* (2012) for a survey of works that consider the problem of dynamic sensor selection under some diagnosability constraint in discrete event systems. It was shown in *Yoo and Lafortune* (2002) that finding a minimal observable event set such that the diagnosability property holds is an NP-complete problem. The same was shown for the DES properties of normality and observability, properties relevant to supervisory control of DESs. This work is most similar to that of *Thorsley and Teneketzis* (2007) and *Cassez and Tripakis* (2008), both of which have considered the problem of dynamic diagnosability.

In *Thorsley and Teneketzis* (2007), the optimal dynamic diagnosability problem is considered for each of: acyclic timed automata, acyclic untimed automata, and general untimed automata. In each case, the goal is to find an optimal controller according to a cost function on sensor activations. Both the logical and stochastic cases are considered. A solution is obtained by first defining an appropriate information state and then using dynamic programming to obtain an optimal solution.

In *Cassez et al.* (2007a), the solution concept of most permissive observer (MPO) is introduced, which the authors use as a basis for optimization according to a non-discounted numerical cost criterion in *Cassez et al.* (2007b); *Cassez and Tripakis* (2008). Instead of an information state based approach, the authors use results from

game theory (particularly safety games) and games on graphs. The same approach is also applied to the problem of opacity in *Cassez et al.* (2009).

Another paper that considers the dynamic diagnosability problem is *Wang et al.* (2010). That paper first defines feasible and implementable controllers over language based partitions and then proceeds to define a paricular class of language based partitions called window partitions. The paper provides an algorithm for obtaining minimal solutions to both the centralized and distributed dynamic diagnosability problems, where minimality is defined over the space of feasible and implementable controllers for a particular window partition. Better solutions can be obtained by taking finer partitions, at the cost of additional computation time.

We also mention *Wang et al.* (2009), which defines the state disambiguation problem and the extended specification, both used in this work. They provide an algorithm for solving the minimal sensor activation problem on-line. Other than diagnosability, it has also been shown that the property of observability can be mapped to state disambiguation in *Wang et al.* (2007). Earlier versions of the results presented in Chapter II appear in *Dallal and Lafortune* (2010, 2011a,b).

1.3.2 Vehicle Control

Three common approaches to the problem of vehicle control include: the computation of maximally controlled invariant sets; mapping the problem to that of scheduling; and abstraction/symbolic models. Among approaches falling in the first category include, e.g., (*Hafner and Del Vecchio*, 2011; *Verma and Del Vecchio*, 2011). By explicitly computing the *capture set*, or set of states from which it is not possible to guarantee avoidance of the unsafe states, these approaches naturally satisfy safety, non-deadlockingness and maximal permissiveness, and can deal with sources of uncontrollability and also with measurement uncertainty. However, such approaches typically make assumptions such as convexity or order preserving dynamics, without

which they do not scale well to systems with multiple dimensions. See also (*Tomlin et al.*, 2003) for an example involving a flight management system. Scheduling approaches work by allocating time intervals during which the vehicles can be inside the intersection.

The scheduling problem is generally NP-hard but takes polynomial time in the special case where all jobs require the same processing time. Reducing the vehicle control problem to the polynomial-time scheduling case amounts to either an assumption of certain symmetries in the vehicle control problem set-up, or a problem relaxation where such symmetries are not satisfied. Approaches in this category include (*Colombo and Del Vecchio*, 2012), its extension to the case of dynamics with disturbances, (*Bruni et al.*, 2013), and its extension to the case of uncontrolled vehicles, *Ahn et al.* (2014). To our knowledge, these methods have not been extended to the case of measurement uncertainty. Furthermore, the assumption of mutual exclusiveness of the intersection's use is restrictive, as it precludes vehicles on common or non-intersecting trajectories (e.g., in the case of right turns) from utilizing the intersection simultaneously. Another approach is to pre-compute fail-safe maneuvers as in (*Kowshik et al.*, 2011), or evasion plans as in (*Au et al.*, 2012). These last approaches deal with some types of environmental uncertainty, but do not guarantee maximal permissiveness.

Finally, abstraction based methods work by mapping the continuous system model and specifications to a finite model and solving for a supervisor on the finite model, in such a way that the obtained supervisor can be used on the original (continuous) system, while preserving safety and non-deadlocking properties. Work in this domain includes (*Alur et al.*, 2000; *Daws and Tripakis*, 1998) in the context of verification / model checking, as well as (*Colombo and Del Vecchio*, 2011a,b; *Colombo and Girard*, 2013), which makes use of differential flatness of dynamical systems to construct abstractions with provable errors bounds. Our work is most closely related to that of

(*Girard et al.*, 2010; *Pola and Tabuada*, 2009; *Zamani et al.*, 2012; *Camara et al.*, 2011), which construct symbolic models that satisfy simulation or alternating simulation relations with the original system. In particular, this work also makes use of alternating simulation relations, and variations thereof.

Besides the language based specifications of supervisory control theory, system specifications can also be formulated using temporal logics such as Linear Temporal Logic (LTL). Linear temporal logic is a language for specifying system behavior which consists of boolean logic and a few "temporal" operators such as Next, Always, Eventually, and Until. LTL can be used to specify allowable sequences of states in a compact way. This approach is used in *Belta et al.* (2007), where the authors use a discretization based on triangularization for the problem of robot control. In *Alur et al.* (2000), the authors characterize classes of hybrid systems in which it is possible to obtain finite discrete abstractions for the purpose of deciding formulas expressed in LTL or Computational Tree Logic (CTL) (a temporal specification language similar to LTL). A particular class of LTL formulas called General Reactivity (GR(1)) formulas are defined in *Piterman et al.* (2006). These formulas place constraints on the initial values, transitions, and goals of the system and the environment. The authors pose the question: do the constraints on the environment imply the specifications on the system. If so, they show that the complexity of translating the formula to an automaton is polynomial in the size of the formula (as oppose to doubly exponential in the general case).

1.4 Solution Methodology & Contributions

1.4.1 Dynamic Diagnosability

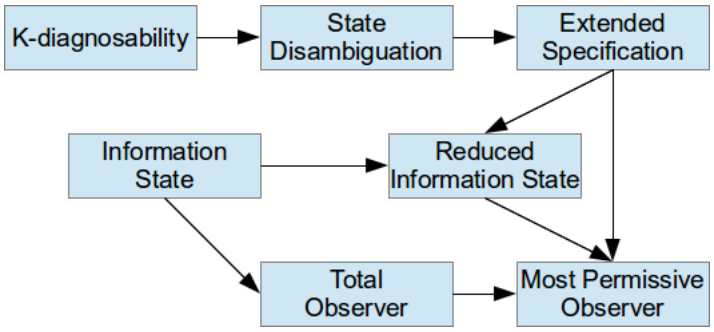


Figure 1.2: A summary of the approach to the construction of the MPO. The K-Diagnosability problem is mapped to the state disambiguation problem, for which the extended specification is computed. An appropriate information state is defined and the TO, containing all admissible controllers, is constructed over the space of information states. By proving a monotonicity property over the extended specification, we are able to reduce the information state. Finally, the MPO is constructed over the space of reduced information states, obtained as a sub-automaton of the TO, and using the extended specification to determine the safety of control actions and reduced information states.

Our solution methodology for the dynamic diagnosability problem is depicted in Fig. 1.2. We begin by providing a more formal behavioral definition of the MPO, the structure which encapsulates the set of all safe controllers. Specifically, we define the MPO as a deterministic bipartite automaton defined over two types of states, called Y and Z states. The MPO has transitions from Y to Z states, labelled with sensor activation decisions in $\Gamma = \{\gamma \in 2^E : E_o \subseteq \gamma \subseteq E_o \cup E_s\}$, and transitions from Z to Y states, labelled with events (corresponding to observations). Now remark that a run can be considered a string in the set $(\Gamma E)^*$. The MPO therefore maps each run to at most one Y state. Let this mapping be denoted by the partial function $\Delta : R \to Y$. We say that the MPO allows controller C if, for every run $\rho \in R$ such that $\Delta(\rho)$ is defined, there is a transition from $\Delta(\rho)$ in the MPO with label $C(\rho)$. The MPO is characterized by the property that a controller must be safe if and only if it is allowed

by the MPO.

With the above behavioral definition of the MPO, the remainder of the work therefore consists of precisely defining what the set of Y and Z states is, what the safe control actions are from any given Y state, and how to efficiently construct the MPO. To this end, we begin by defining an appropriate notion of information state for this problem, and let the set of such states be denoted by I. Specifically, these information states satisfy two critical properties:

- The information state is uniquely determined by the sequence of control actions and observations (i.e., the run).
- The set of safe control actions after any given run is uniquely determined by the reached information state.

With these properties, we can then let $Y = I$, and $Z = I \times \Gamma$, where Γ is the set of feasible control actions. The reason for the definition of Z as $I \times \Gamma$ is that transitions from Z states to Y states in the MPO correspond to event observations, and the set of possible event observations is a subset of the monitored events. Thus, any Z state must "remember" the previous control action in order to determine the outgoing transitions from this Z state.

We proceed to define the TO, which has the same transition structure as the MPO, but which allows *all* controllers, not merely the safe ones. It follows that the MPO will be a sub-automaton of the TO. In practice, there may be information states that cannot be produced by any run, and hence only a subset of the set of Y and Z states will actually be reachable in the TO. Furthermore, some runs may occur only under unsafe controllers, and hence the set of Y and Z states present in the MPO will be smaller still. The next step, therefore, is to determine which Y and Z states (among those in the TO) will in fact be present in the MPO. We call these remaining Y and Z states safe.

To determine this set of safe Y and Z states, we begin by mapping the K-diagnosability problem to that of *state disambiguation*. Informally, given automaton $G^{sd} = (X^{sd}, \Sigma, f^{sd}, x_0^{sd})$ and subset of monitorable events $\Sigma_0 \subseteq \Sigma$, the dynamic state disambiguation problem consists of finding a dynamic controller (which also chooses sets of events to monitor) such that it is always possible to differentiate between pairs of states in a given specification $T_{\text{spec}} \subseteq X^{sd} \times X^{sd}$. That is, the controller ensures that the system's information state never contains both states of some pair $(x_1, x_2) \in T_{\text{spec}}$. We show that the K-diagnosability problem is equivalent to an appropriately defined state disambiguation problem. This in turn allows us to make use of the *extended specification* of the state disambiguation problem. In words, the extended specification T_{spec}^e contains all state pairs that must not be confused because such confusion would make it impossible to ensure that the system does not confuse a pair of states in the specification T_{spec}, not even by turning on all available sensors from that point onwards. We then prove that a Y or Z state is safe if and only if satisfies the extended specification, a condition which can be verified in time quadratic in the size of G. Using this quadratic time test, we can then construct the MPO through a depth-first search over the states of the TO, leaving only the safe Y and Z states in the final MPO.

Finally, we prove a monotonicity property on the extended specification that, as a corollary, allows us to reduce the set I of information states to a smaller set of *reduced information states*, without losing any information necessary for diagnosis. Thus, the MPO can be constructed over the smaller space of reduced information states, yielding a lower space complexity.

Our contributions are as follows. By defining the information state as we have, we reduce the space complexity of the MPO from $O(2^{|X|^2 \cdot K \cdot 2^{|E|}})$ in *Cassez and Tripakis* (2008) to $O(2^{|X| \cdot (K+2)} 2^{|E|})$. Furthermore, the monotonicity property on the extended specification and the resulting reduction of the information state reduces this com-

plexity to $O(2^{|X|}(K+2)^{|X|}2^{|E|})$, yielding a complexity that is polynomial in K, rather than exponential in K. Finally, the use of the extended specification allows for an efficient quadratic time test for determining the safety of a control decision from any given information state. This potentially allows for a minimal solution to be computed on-line, simply by taking a minimal control decision from each information state (all such controllers will be minimal but not all minimal controllers will have this form).

1.4.2 Vehicle Control

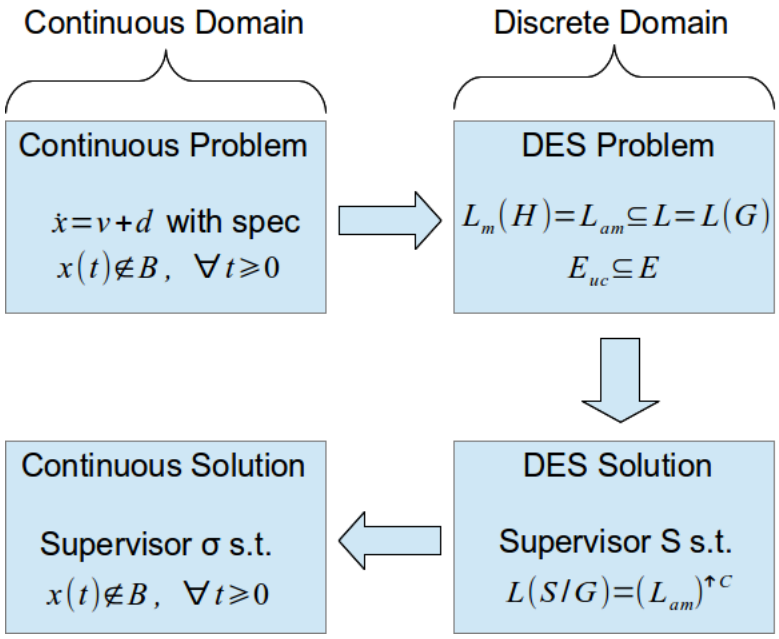


Figure 1.3: A depiction of the solution method in the case of perfect measurement. The continuous system is discretized in time and space and a DES abstraction G is defined over the discrete state space. The problem specifications are translated to a sub-automaton H, and a supervisor for the abstracted system is obtained by solving problem BSCP-NB. Finally, the continuous domain supervisor σ is obtained from the supervisor S of the DES domain.

Our solution methodology for the vehicle control problem in the case of perfect measurement is depicted in Fig. 1.3. The first step consists of translating the system model (dynamics) and specifications (safety and non-deadlockingness) to the DES

domain. This is achieved by discretizing the continuous time system in space and time, a technique called *abstraction*. We begin by discretizing the system in time with parameter τ, so that vehicles choose control actions at times $0, \tau, 2\tau, \ldots$ and hold those control decisions for time τ. This yields the discrete time system:

$$x_{k+1} = x_k + u_k + \delta_k \tag{1.1}$$

with $x_k = x(k\tau)$, $u_k = v(k\tau)\tau$, and $\delta_k = \int_{k\tau}^{(k+1)\tau} d(t)dt$. Calling $U = V\tau$ and $\Delta = D\tau$, we have that $u \in U$ and $\delta \in \Delta$. Recall that a certain subset of the vehicles is uncontrolled. To represent this, we write $u = (u_c, u_{uc})$, for any $u \in U$, where u_c is the control action of the controlled vehicles and u_{uc} is the control action of the uncontrolled vehicles. We also write $U = U_c \times U_{uc}$, and similarly, $V = V_c \times V_{uc}$.

Next, we discretize the system in space with a parameter of $\tau\mu$, yielding a lattice of states $\tilde{Q}$ with spacing $\tau\mu$. Each continuous state $x \in X$ is therefore mapped to a discrete state $q \in \tilde{Q}$ through the function $\ell : X \to \tilde{Q}$. Fig. 1.4 depicts the real time system operation.

As the next step, we define a DES abstraction $G = (Q, E, \psi, q_0, Q_m)$ over the set of discrete state $\tilde{Q}$ to capture the dynamics of the discrete time system of Eq. (1.1). The events of G are U_c (for the actions of the controlled vehicles), U_{uc} (for the actions of the uncontrolled vehicles), and W (which is a discretization of the set D and represents the "actions" of the disturbance). The set of controllable events is U_c and the set of uncontrollable events is $U_{uc} \cup W$. Each of these event classes forms one of three "layers" of the transition function ψ, so that $\mathcal{L}(G) \subseteq (U_c U_{uc} W)^*$ (n.b.: this requires the addition of two layers of "intermediate states", Q_{I1} and Q_{I2}). See Fig. 3.3 for a depiction of the transition function ψ. The transition function ψ satisfies the property that, for any $q \in \tilde{Q}$, $u_c \in U_c$, and $q' \in \tilde{Q}$, there exist $u_{uc} \in U_{uc}$ and $w \in W$ such that $q' = \psi(q, u_c u_{uc} w)$ if and only if there exist $x \in X$, $\delta \in \Delta$, and $x' \in X$ such that $q = \ell(x)$, $q' = \ell(x')$, and $x' = x + u + \delta$, where $u = (u_c, u_{uc})$. Finally, $Q_m = \{q_m\}$,

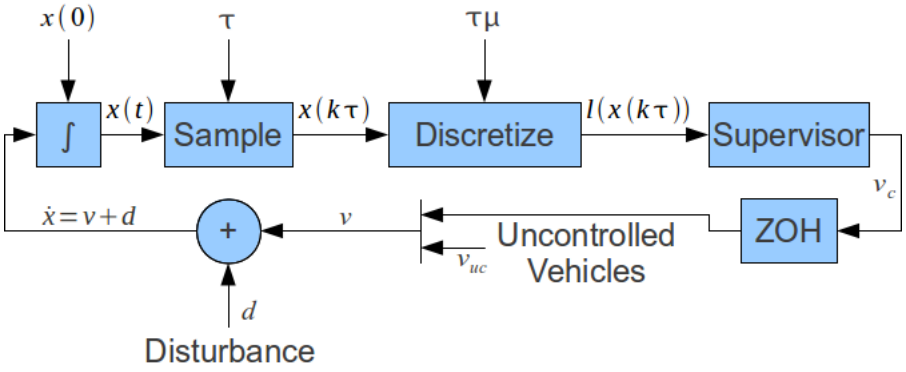


Figure 1.4: A depiction of the real time system operation of the vehicle control system. The state $x(t)$ is sampled at times $0, \tau, 2\tau, \ldots$, yielding $x(k\tau)$. This sample is then discretized through the function $\ell(\cdot)$ to a lattice point in the set $\tilde{Q}$. The discrete state $\ell(x(k\tau))$ is sent to the supervisor, which allows a subset of the available control actions. One of these control actions is chosen by the vehicles, and held for the following interval of length τ (i.e., a zero-order hold). Finally, the chosen control action v_c, in addition to the actions of the uncontrolled vehicles v_{uc} and the effect of the disturbance d determine the system trajectory.

where $\ell(x) = q_m$ for any $x \in X$ where all the vehicles have crossed the intersection.

This completes the translation of the system model from the continuous domain to the DES domain. The next step is to translate safety and non-deadlocking specifications to the DES domain as well. To translate the safety requirement, we define a transition from q to $q' = \psi(q, u_c u_{uc} w)$ in G as safe if all trajectories of the continuous time and space system corresponding to this discrete transition do not cross the bad set. The translation of the non-deadlocking specification is achieved through the DES marking: the set of marked states $Q_m = \{q_m\}$ corresponds to the continuous states where all the vehicles have crossed the intersection. Requiring that the DES system be able to reach the state q_m therefore implies that the vehicles will cross the intersection in the original, continuous, domain system. We therefore define a second automaton H, which is the sub-automaton of G containing only safe transitions. Solving problem BSCP-NB with marked legal language $L_{am} = \mathcal{L}_m(H)$, G, and uncontrollable event set $E_{uc} = U_{uc} \cup W$ therefore yields a maximally permissive

safe and non-deadlocking supervisor S for the DES domain problem. The continuous domain supervisor σ is obtained from S through $\sigma(x(k\tau)) = S(\ell(x(k\tau)))$, for $k \in \mathbb{N}$, and control actions are held for the following interval of length τ.

It remains to show that the supervisor σ will be safe, non-deadlocking, and maximally permissive. To this end, we define two types of relations between systems and their abstractions: the state reduction and the exact state reduction. We show that, when an abstraction is a state reduction of some initial system and the safety and non-deadlocking specifications of the initial system are translated to the abstract domain as *induced specifications*, then the supervisor for the initial system obtained by following the procedure of Fig. 1.3 will be safe, non-deadlocking, and maximally permissive among the class of memoryless supervisors, given the spatial discretization. Furthermore, when the abstraction is an *exact* state reduction of the initial system, the resulting supervisor for the initial system will be safe, non-deadlocking, and maximally permissive among the class of all controllers (given the spatial discretization), not merely memoryless ones.

Finally, we show that the abstraction G is indeed a state reduction of the continuous domain vehicle control system, and that the specifications given by sub-automaton H are in fact the induced safety and non-deadlocking specifications. Thus, we can invoke the preceding theorem to show that the obtained continuous domain supervisor σ is indeed safe, non-deadlocking, and maximally permissive. Furthermore, we also show that, when the bounds $d_{\min}$ and $d_{\max}$ on the disturbance are integer multiples of μ (recall that μ is the discretization parameter for the velocities V), then G is an *exact* state reduction of the continuous domain vehicle control system.

We now turn attention to the case of imperfect measurement. Our solution methodology for this case is depicted in Fig. 1.5. The idea is very similar to that of the case of perfect measurement, except that, in this case, we construct a DES automaton that is an abstraction of the *prediction-correction estimator* of the con-

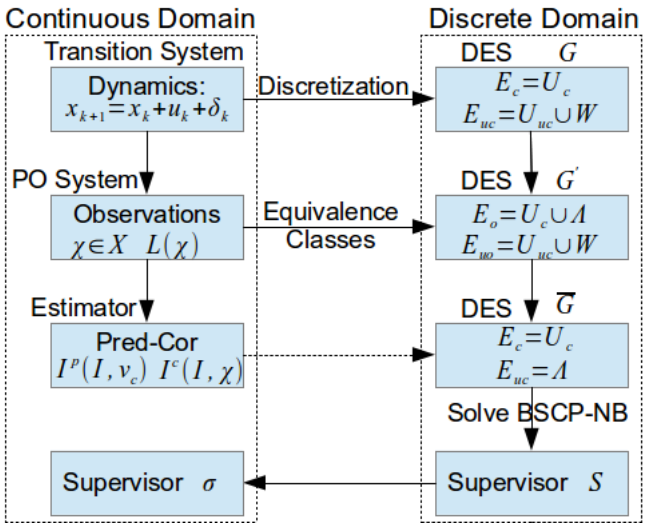


Figure 1.5: The solution method. Given the time discretized system of Eq. (1.1) with the space discretization given by $\ell(\cdot)$, constituting system S_b, we can construct DES G that is a state reduction of this system. Given system S_b with measurement uncertainty given by $L(\cdot)$, constituting partially observed system S_b', we can construct DES G' that models the measurement uncertainty of $L(\cdot)$ by partitioning the set of measurements X into equivalence classes Λ. Given the estimator $\overline{S}_b$ of partially observed system S_b', we can construct DES $\overline{G}$ that is a state estimate reduction of $\overline{S}_b$. Furthermore, when $e_{\max} = k\mu\tau/2$ for some $k \in \mathbb{N}$, $\overline{G}$ can be obtained as the observer of G'. A DES supervisor S is then computed by solving problem BSCP-NB, from which a continuous domain supervisor σ is obtained.

tinuous system. In this setting, the actions of the uncontrolled vehicles and of the disturbance are not directly observed. Instead, we obtain information about them through measurements χ, taken at times $0, \tau, 2\tau, \ldots$. Given some maximum measurement error $e_{\max}$, we define the function $L : X \to 2^X$ by:

$$L(\chi) = [\chi - \mathbf{1}e_{\max}, \chi + \mathbf{1}e_{\max}], \tag{1.2}$$

where $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^n$ and, for any $a, b \in \mathbb{R}^n$, $[a, b] := \{x \in \mathbb{R}^n : a_i \leq x_i \leq b_i, i = 1, \ldots, n\}$ denotes a box. Thus, for any measurement $\chi \in X$, $L(\chi)$ consists of the set of states that are consistent with measurement χ. Recall that V_{uc} is the set of actions of the uncontrolled vehicles (i.e., velocities), and $\Delta = D\tau$ is the set of possible

disturbances for the discrete time system of Eq. (1.1). With L defined as above, we can define a prediction-correction estimator as the two functions $I^p : 2^X \times V_c \to 2^X$ and $I^c : 2^X \times X \to 2^X$ (n.b.: X is both the set of possible vehicle positions and the set of possible measurements), given by:

$$I^p(I, v_c) = \bigcup_{x \in I} \bigcup_{v_{uc} \in V_{uc}} \bigcup_{\delta \in \Delta} (x + v\tau + \delta) \tag{1.3}$$

$$I^c(I, \chi) = I \cap L(\chi), \tag{1.4}$$

where $v = (v_c, v_{uc})$. Thus, if $I \subseteq X$ is the current best estimate of the system state at some time t and control action $v_c \in V_c$ is taken then $I^p(I, v_c)$ predicts the set of states that the system could be in at time $t + \tau$. Similarly, if I is the current estimate of the system's state and measurement χ is obtained then the corrected estimate is given by $I^c(I, \chi)$. The prediction-correction estimator given by Eqs. (1.3) and (1.4) define a new transition system defined over state estimates. Specifically, the partial transition function of this system is some subset of $2^X \times V_c \times 2^X$, with transition (I, v_c, I') present in this system if there exists some $\chi \in X$ such that $I' = I^c(I^p(I, v_c), \chi)$. Let this transition system defined over state estimates be denoted by $\overline{S}_b$. The next step is to construct a suitable DES abstraction of $\overline{S}_b$.

For the vehicle control problem under consideration, this is achieved by modifying the DES abstraction G that was constructed in the case of perfect information and then taking the *observer* of the modified abstraction. The DES abstraction G is modified by partitioning the set of measurements X into a set of equivalence classes, given the discretization function $\ell(\cdot)$. Mathematically, $\chi_1 \equiv_o \chi_2 \Leftrightarrow \ell(L(\chi_1)) = \ell(L(\chi_2))$. Thus, two measurements are grouped in the same equivalence class if the set of discrete states consistent with both measurements are the same. Next, we let $[\chi]$ denote the equivalence class of measurement $\chi \in X$ and Λ be the set of such

equivalence classes. The set Λ is then used as a set of discrete measurement events for the modified abstraction G'. The set of events Λ is taken to be the first layer of the four layer transition function ψ' of G', so that $\mathcal{L}(G') \subseteq (\Lambda U_c U_{uc} W)^*$. The state of G' reached after any string in the set $(U_c U_{uc} W \Lambda)^*$ is some element of $\tilde{Q}'$, which is a copy of the set of discrete states $\tilde{Q}$. For any $\lambda \in \Lambda$, $q \in \tilde{Q}$, and $q' \in \tilde{Q}'$ that is the copy of q, we have that:

$$\psi'(q', \lambda) := \begin{cases} q, & \text{if } (\exists \chi \in X : [\chi] = \lambda)[L(\chi) \cap \ell^{-1}(q) \neq \emptyset] \\ \text{undefined}, & \text{else} \end{cases} \tag{1.5}$$

In words, there is a transition from the copy of q to q itself upon measurement event λ if and only if there exists some (continuous) measurement χ whose equivalence class is λ and some continuous state x mapping to discrete state q such that x is consistent with χ. Intuitively, the measurement event λ can not occur when in state q if the above condition does not hold. If the condition does hold, then the act of measuring does not change the system's physical state, so there is a transition between q' and q. Because the measurement events Λ are observed but cannot be chosen by the system, we take these events to be observable but uncontrollable. On the other hand, the events of uncontrolled vehicles U_{uc} and of the disturbance W are not directly observed by the system, and as such are taken to be unobservable (as well as uncontrollable). The set of control events U_c are taken to be both controllable and observable. This is summarized in Fig. 1.6.

Finally, let $\overline{G} = Obs(G')$, the observer of DES G'. This observer is a new automaton whose states are state estimates of G', obtained by performing a determinization with respect to the unobservable events. Its event set is only the observable events, namely the (controllable) events U_c and the (uncontrollable) events Λ. It can be shown that $\overline{G}$ is effectively a prediction-correction estimator. As in the perfectly measured case, we proceed to translate safety and non-deadlocking requirements to

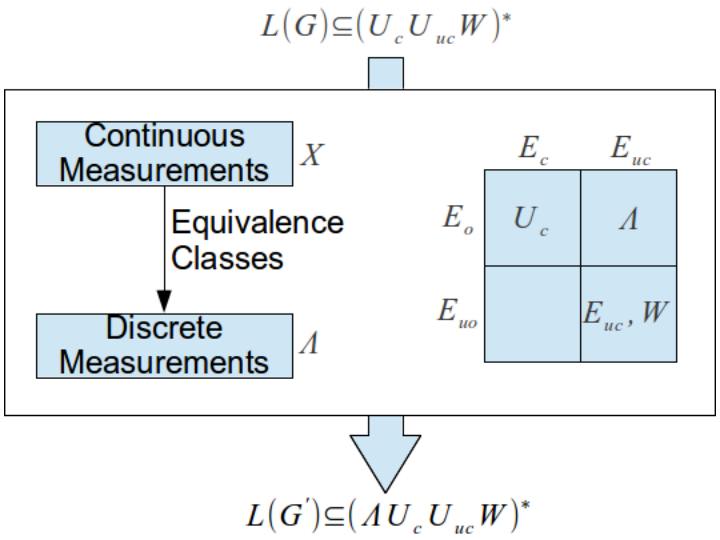


Figure 1.6: A depiction of the process by which the modified DES abstraction G' is constructed from DES G. The set of continuous measurements X is partitioned into a set of equivalence classes with respect to the function $\ell(L(\cdot))$, yielding the set of equivalence classes Λ representing measurement events. The events of U_c are classified as controllable and observable, the events of U_{uc} and W are classified as uncontrollable and unobservable, and the events of Λ are classified as uncontrollable but observable. With four classes of events, the language of G' becomes $\mathcal{L}(G') \subseteq (\Lambda U_c U_{uc} W)^*$.

the DES domain, given by a sublanguage of $\mathcal{L}(\overline{G})$, following which we solve problem BSCP-NB and translate the obtained supervisor S from the DES domain to the continuous domain, yielding supervisor σ.

As before, it remains to prove that σ satisfies the safety and non-deadlocking specifications of the continuous time system, and to characterize the class of supervisors over which σ is maximally permissive. To this end, we define the analogous versions of the state reduction and the exact state reduction, but for the case of imperfect measurement: the state estimate reduction and the exact state estimate reduction. The analogous theorems show that, when some abstraction is a state estimate reduction of a prediction-correction estimator, and the safety and non-deadlocking specifications of the initial system are translated to the abstract domain as induced specifications, then the supervisor for the initial system obtained by following the procedure of Fig. 1.5 will be safe, non-deadlocking, and maximally permissive among the class

of memoryless supervisors, given the spatial discretization. Furthermore, when the abstraction is an exact state estimate reduction of the initial system, the resulting supervisor for the initial system will be safe, non-deadlocking, and maximally permissive among the class of all controllers (given the spatial discretization), not merely memoryless ones.

Finally, we show that, when the maximal measurement error $e_{\max}$ is an integer multiple of $\mu\tau/2$, then the observer $\overline{G}$ will indeed be a state estimate reduction of the prediction-correction estimator $\overline{S}_b$. Furthermore, if bounds on the disturbance $d_{\min}$ and $d_{\max}$ are also multiples of $\mu\tau$ (i.e., for the same condition under which the abstraction G of the perfectly measured case is an exact state reduction), then $\overline{G}$ will be an *exact* state estimate reduction of the prediction-correction estimator $\overline{S}_b$. On the other hand, if $e_{\max}$ is not an integer multiple of $\mu\tau/2$, then the above procedure does not yield a state estimate reduction. In such a case, it is still possible to obtain an abstraction that is a state estimate reduction, but only by directly constructing an abstraction of the prediction-correction estimator.

Our contributions are as follows. We showed how to construct DES abstractions for systems with environmental uncertainty by discretizing the state space, using uncontrollable events to model sources of environmental uncertainty and, in the case of imperfect measurement, using observable but uncontrollable events to model measurement uncertainty. We also showed how to translate safety and marking specifications defined over a continuous state space to the DES domain, yielding a language based specification. Finally, we defined new relations between systems and their DES abstractions, allowing for supervisors to be computed using a "abstract-solve-translate" method and characterizing the class of supervisors over which the result will be maximally permissive.

CHAPTER II

Dynamic Diagnosability

2.1 Abstract

We consider the problem of dynamic sensor activation for fault diagnosis of discrete event systems modeled by finite state automata under the constraint that any fault must be diagnosed within no more than $K + 1$ events after its occurrence, a property called K-diagnosability. We begin by defining an appropriate notion of information state for the problem and defining dynamic versions of the projection operator and information state evolution. We continue by showing that the problem can be reduced to that of state disambiguation. Then we define the most permissive observer (MPO) structure that contains all the solutions to the problem, and we prove results showing that maintaining the K-diagnosability property is equivalent to satisfying the extended specification of the state disambiguation problem. We then prove a monotonicity property of the extended specification, and show that this allows us to reduce our information state, which in turn allows us to significantly reduce the complexity of our solution. Putting all of our results together, we obtain a MPO with a size complexity of $O(2^{|X|}(K+2)^{|X|}2^{|E|})$, compared with $O(2^{|X|^2 \cdot K \cdot 2^{|E|}})$ for the previous approach, where X and E are respectively the sets of states and events of the automaton to diagnose. Finally, we provide an algorithm for constructing the most permissive observer and demonstrate its scalability through simulation.

2.2 Introduction

The problem under consideration in this work is that of dynamic fault diagnosis for discrete event systems modelled by finite state automata. We assume that there are sensors capable of detecting event occurrences for a subset of the events of the automaton model of the system. Among those events that are monitorable, it is further assumed that there is a subset whose sensors are costly to operate. This may be because of limited availability of energy or bandwidth, out of a desire to minimize communication for security reasons, or for any other reason. We use the sensor outputs and system model to diagnose past fault occurrence. In this work, we consider the K-diagnosability property, which stipulates that the occurrence of a fault must be determined with certainty within no more than $K + 1$ events after its occurrence. It is assumed that we do not have sensors capable of detecting these fault events directly. Thus, the structure of the problem presents a trade-off: if sensors are turned on too infrequently, we may fail to diagnose the fault in time; if sensors are turned on too frequently, fault diagnosis will be needlessly costly. The problem is dynamic because we assume that sensors can be turned on or off at different points in the system's execution. A controller in this problem is (in the most general sense) a function mapping histories of past sensor activations (i.e., the control decisions) and observed events to a set of sensor activations.

Related literature in the context of *static* sensor selection problems in discrete event systems include e.g., *Haji-Valizadeh and Loparo* (1996), *Jiang et al.* (2003) and *Yoo and Lafortune* (2002). The sensor selection problem is of particularly great importance in wireless sensor networks due to energy limitations. For a survey of work in this domain, see *Rowaihy et al.* (2007). See *Zaytoon and Sayed Mouchaweh* (2012) for a survey of works that consider the problem of dynamic sensor selection under some diagnosability constraint in discrete event systems. In particular, we describe *Thorsley and Teneketzis* (2007); *Cassez and Tripakis* (2008); *Wang et al.*

(2010) here. In *Thorsley and Teneketzis* (2007), the emphasis is on finding a single dynamic controller that solves the diagnosability problem optimally according to a discounted numerical cost criterion. The authors use an information state based approach in combination with dynamic programming. In *Cassez et al.* (2007a), the solution concept of most permissive observer (MPO) is introduced, which the authors use as a basis for optimization according to a non-discounted numerical cost criterion in *Cassez et al.* (2007b); *Cassez and Tripakis* (2008). Instead of an information state based approach, the authors use results from game theory (particularly saftey games) and games on graphs. The same approach is also applied to the problem of opacity in *Cassez et al.* (2009). Finally, *Wang et al.* (2010) considers the decentralized version of the dynamic diagnosis problem. The authors use a "window-based partition" approach to obtain polynomial time algorithms for computing solutions in the centralized and decentralized cases.

In this work, we provide a new information state based characterization of the MPO structure originally defined in *Cassez et al.* (2007a), an approach similar to that used in *Thorsley and Teneketzis* (2007). We prove that our information state is sufficient to fully determine the set of all control decisions (i.e., sensor activations) that maintain the K-diagnosability property. Our goals in providing a new information state based characterization of the MPO are three-fold: (i) to obtain a more readily interpretable solution; (ii) to better study informational properties of the MPO; and (iii) to have a method that could more easily be adapted to other dynamic optimization problems in discrete event systems. We show that the fault diagnosis problem under consideration can be reduced to a state disambiguation problem, which we use to prove a number of monotonicity properties about the MPO. Notably, we show that satisfying the K-diagnosability property is equivalent to satisfying the extended specification of a state disambiguation problem. Finally, we prove a monotonicity property of the extended specification, and show how this allows us to reduce our

information state, without losing any "useful" information for the purpose of diagnosis. Putting these results together, we present an algorithm for constructing the MPO. The MPO can then be used as the basis for solving an optimal dynamic sensor activation problem, as is done in *Cassez and Tripakis* (2008); *Cassez et al.* (2007b) or *Thorsley and Teneketzis* (2007).

The MPO constructed in this work improves upon that of *Cassez and Tripakis* (2008) in four ways. First, by using a single automaton to track both faulty and non faulty executions rather than the product of two automata, we obtain a space complexity for our MPO that is exponential in $|X|$, rather than exponential in $|X|^2$, where X is the state space of the automaton to diagnose. Second, by defining an information state at the outset and then introducing control decisions on these information states, we avoid performing a determinization after introducing control decisions. This allows us to obtain a space complexity for our MPO that is exponential in $|E|$, rather than doubly exponential in $|E|$, where E is the set of events of the automaton to diagnose. Third, by mapping the problem to that of state disambiguation and making use of the extended specification, we allow for an on-line construction of a minimal solution when $|E|$ is small. Finally, by proving a monotonicity property on the extended specification, we reduce the number of distinct information states, resulting in an MPO that has polynomial size in K, rather than exponential in K. The final size complexity of our MPO is $O(2^{|X|}(K+2)^{|X|}2^{|E|})$, compared with $O(2^{|X|^2 \cdot K \cdot 2^{|E|}})$ in *Cassez and Tripakis* (2008). Preliminary versions of some of the results in this work have appeared in *Dallal and Lafortune* (2010), *Dallal and Lafortune* (2011a), and *Dallal and Lafortune* (2011b).

The remainder of this work is organized as follows. In Sect. 2.3, we formally define the problem we wish to solve. In Sect. 2.4, we provide definitions related to our notion of information state and define the total observer, which contains all admissible controllers. In Sect. 2.5, we present the state disambiguation problem,

show how the K-diagnosability problem can be mapped to it, and use this mapping to construct the most permissive observer by pruning the total observer. In Sect. 2.6, we define the extended specification and prove that K-diagnosability is maintained if and only if the extended specification is satisfied. In Sect. 2.7, we prove the monotonicity property on the extended specification that allows us to reduce our information state. In Sect. 2.8, we present an algorithm for constructing the MPO, give its running time, and provide experimental results demonstrating the scalability of the algorithm in practice. Finally, we conclude in Sect. 2.9. Appendix 1 contains algorithms and their running times for computing the extended specification and the reduced unobservable reach, which are used in the construction of the MPO. Appendix 2 contains the proofs of some results.

2.3 Problem Formulation

We begin by defining the dynamic diagnosability problem more precisely. Assume that the system to be diagnosed is modeled by a *deterministic* finite state automaton. We use the standard deterministic model that has been adopted in the literature on supervisory control *Ramadge and Wonham* (1989) and diagnosis *Sampath et al.* (1995) in discrete event systems. Specifically, let $G = (X, E, f, x_0)$, where X is the set of states, E is the set of events, $f : X \times E \to X$ is a partial transition function, and x_0 is the initial state. Let E^* denote the set of all finite length strings of events in E. The transition function f is extended from the domain $X \times E$ to $X \times E^*$ recursively: $f(x, es) := f(f(x, e), s)$ for all $s \in E^*$. We denote by $\mathcal{L}(G, x)$ the set of all strings $s \in E^*$ that can occur through f when starting from x. For brevity, we denote by $\mathcal{L}(G) := \mathcal{L}(G, x_0)$ the language of the system and we define $f(s) := f(x_0, s)$ for all $s \in E^*$. The set of events E is partitioned into four mutually exclusive categories: $E = E_o \cup E_s \cup E_{uo} \cup E_f$, where E_o is the set of freely monitorable events (events for which we have zero cost sensors), E_s is the set of costly monitorable events (events

for which we have costly sensors), E_{uo} is the set of non-faulty unobservable events (non-faulty events for which we do not have sensors), and $E_f = \{e_f\}$ contains the fault event whose occurrence we would like to diagnose. The generalization of the results to the case of multiple fault events and / or types is straightforward.

Our goal is to dynamically diagnose the occurrence of the fault event e_f within no more than $K+1$ events after the occurrence of the fault. Note that this is $K+1$ events of any kind, whether we can observe the events or not. This is referred to as the *K-diagnosability property*. A controller for this problem is a function that chooses a set of events to monitor. This set of events remains fixed until an observation is made, at which point a new set of events to monitor can be chosen (i.e., a new control decision). This naturally produces an alternating sequence of control decisions and event observations, which we call a run (Def. II.1); these runs constitute the domain for controllers (Def. II.2). With these definitions, we can define a function mapping executions to observations (Def. II.3) and hence formally define the K-diagnosability property (Def. II.4). We end this section by defining the problem we wish to solve (Prob. II.5).

Definition II.1 (Run)**.** A run ρ of length n is defined as a sequence $C_0, e_0, \ldots, C_{n-1}, e_{n-1}$ of *control decisions* or *sensor activations* (the C_i's, which are subsets of events to monitor) and observed events (the e_i's). Since the events are observed, they must be among the monitored events. That is, $e_i \in C_i$, for all $i = 0, \ldots, n-1$. On the other hand, the strict alternation of control decisions and observed events reflects the assumption that control decisions are only changed upon the observance of an event. Denote by R_n the set of runs of length n and by $R = \bigcup_{n=0}^{\infty} R_n$ the set of all runs. Finally, let $\rho(k) = C_0, e_0, \ldots, C_{k-1}, e_{k-1}$ denote the subsequence of ρ of length k. □

Definition II.2 (Admissible Controller)**.** Let $\Gamma = \{\gamma \in 2^E : E_o \subseteq \gamma \subseteq E_o \cup E_s\}$ be defined as the set of admissible control decisions (i.e., Γ is the set of control decisions that monitor all events in E_o and no events in E_{uo} or E_f). Then a controller is defined

as any function $C : R \to \Gamma$ from runs to admissible control decisions. □

For a fixed set of monitored events, it is a trivial task to define the projection of a string. When the set of monitored events changes dynamically along the string's execution, in a way that depends on the particular controller C, it is necessary to define a *controller induced* projection.

Definition II.3 (Controller Induced Projection)**.** Given a controller C, we define $P_C(\rho, s)$ as the string t that is observed when string s occurs after run ρ (the projection t does not include the observed events of ρ). This can be computed as follows:

$$\begin{aligned} P_C(\rho, \varepsilon) &:= \varepsilon \\ P_C(\rho, e) &:= \begin{cases} e & \text{if } e \in C(\rho) \\ \varepsilon & \text{if } e \notin C(\rho) \end{cases} \\ P_C(\rho, es) &:= \begin{cases} e.[P_C(\rho.C(\rho).e, s)] & \text{if } e \in C(\rho) \\ P_C(\rho, s) & \text{if } e \notin C(\rho) \end{cases} \end{aligned} \tag{2.1}$$

For the last case, the first argument of P_C must be updated with the new run. If the event e is not observed then the run does not change. If e is observed, then we produce the new run by concatenating the last control decision and the observed event to ρ. For brevity, we define $P_C(s) := P_C(\rho_0, s)$, where ρ_0 is the empty run (i.e., the unique element of R_0). □

We also define the (static) natural projection $P : E^* \to (E_o \cup E_s)^*$ in the usual way. Equivalently, we can write $P(s) = P_{C_{\text{all}}}(s)$, with C_{all} defined by $C_{\text{all}}(\rho) = E_o \cup E_s$, for all $\rho \in R$. That is, C_{all} is the controller that always monitors all available sensors and $P(s)$ is the projection induced by this controller. We can now formally define the K-Diagnosability property.

Definition II.4 (K-Diagnosability)**.** We recall the standard definition of diagnosability from *Sampath et al.* (1995). Adapted for a fixed K and the context of a dynamic

observer, we say that a system G is K-diagnosable given controller C if there do not exist a pair of strings $s_Y, s_N \in \mathcal{L}(G)$ such that the following three conditions are satisfied:

1. s_Y has an occurrence of the fault event e_f and s_N does not.
2. s_Y has at least $K+1$ events after the fault event e_f.
3. $P_C(s_Y) = P_C(s_N)$, that is, the observed string of events is identical given the controller.

We also say that a system G is K-diagnosable if there exists a controller C such that G is K-diagnosable given controller C. We call such a controller *safe*. □

We now formally define the problem we wish to solve. The following problem definition is an adaptation of the behavioural definition of the most permissive observer found in *Cassez and Tripakis* (2008).

Problem II.5 (Behavioral Definition of the Most Permissive Observer)**.** We would like to find a deterministic bipartite automaton $MPO = (Y \cup Z, \Gamma \cup E, h_{YZ} \cup h_{ZY}, y_0)$ satisfying the following properties:

1. Control decisions are made from Y states: $h_{YZ} \subseteq Y \times \Gamma \times Z$.
2. Observations are made from Z states: $h_{ZY} \subseteq Z \times E \times Y$ and, for any $z \in Z$ and $e \in E$, $h_{ZY}(z, e)$ is defined if and only if there exists some $y \in Y$ and $\gamma \in \Gamma$ such that $h_{YZ}(y, \gamma) = z$ and $e \in \gamma$. This means that, if $z \in Z$ was reached through monitoring decision γ from a previous Y state, then there exists an outgoing transition from z with label e for each $e \in \gamma$.

With MPO's transition function defined in this way, we can map any run $\rho \in R$ to at most one Y state (the converse need not hold). Let the partial function $\Delta : R \to Y$ represent this mapping and define $\Delta(\rho_0) = y_0$, where ρ_0 is the empty run. We say

that controller C is *allowed* by MPO if, for all $\rho \in R$ such that $\Delta(\rho)$ is defined, we have that $h_{YZ}(\Delta(\rho), C(\rho))$ is defined. Notice from property 2) that, if $\Delta(\rho)$ is defined and $h_{YZ}(\Delta(\rho), \gamma)$ is also defined, then $\Delta(\rho.\gamma.e)$ will be defined for all $e \in \gamma$, and will satisfy $\Delta(\rho.\gamma.e) = h_{ZY}(h_{YZ}(\Delta(\rho), \gamma), e)$. From this observation and the fact that $\Delta(\rho_0)$ is defined, it follows by induction that any controller C allowed by MPO will never produce a run ρ such that $\Delta(\rho)$ is undefined. The last property of MPO is therefore as follows:

3. MPO contains exactly the safe controllers: C is allowed by $MPO \Leftrightarrow C$ is safe. □

Such a structure clearly exists. In the worst case, we use a distinct Y state for every possible run in R (in which case Δ will be invertible) and define a transition for $h_{YZ}(y, \gamma)$ if and only if there exists some safe controller C such that $C(\Delta^{-1}(y)) = \gamma$. Defining MPO in this way will result in an infinite structure, however. A procedure for obtaining a finite MPO is given in *Cassez and Tripakis* (2008) and thus the goal of this work is to make use of structural properties of the problem to find a more compact structure that still satisfies all of these properties.

2.4 Towards an Information State

Finding a compact structure for the solution to Problem II.5 requires us to reduce the infinite domain of runs to some finite domain. Hence, the first step is to define an appropriate information state that summarizes the information of the run. We begin by defining the augmented state, augmented transition function, and augmented automaton. Note that the augmented state used here was previously defined in *Cassez and Tripakis* (2008). This work differs in that we do not synchronize these states with those of a copy of the plant with fault events removed. As a consequence, we obtain a

size and time complexity for constructing the MPO that is exponential in $|X|$ rather than exponential in $|X|^2$.

Define the *augmented state* as a pair $(x, n) \in X \times \{-1, 0, 1, \ldots\}$, where, n represents a "count" of the number of events (of all kinds) that have occurred since a first fault event occurred, or -1 if no fault event has occurred. The set of such states is denoted by $X^+ = X \times \{-1, 0, 1, \ldots\}$. The initial augmented state is $x_0^+ = (x_0, -1)$. For any augmented state $x^+ \in X^+$, we let the state and count components be denoted by $S(x^+)$ and $N(x^+)$, respectively, so that $x^+ = (S(x^+), N(x^+))$. We extend this notation to sets by defining $S(U) = \bigcup_{u \in U} S(u)$, for any $U \subseteq X^+$. Next, we define the *augmented transition function* $g : X^+ \times E \to X^+$ on augmented states that is *induced* by the automaton $G = (X, E, f, x_0)$ and the partition of event set E. Formally, for any $u = (x_u, n_u)$ and event e, we have:

- **Case 1:** If $n_u = -1$ and $e \neq e_f$ then $g(u, e) = (f(x_u, e), -1)$.
- **Case 2:** If $n_u = -1$ and $e = e_f$ then $g(u, e) = (f(x_u, e), 0)$.
- **Case 3:** If $n_u \geq 0$ then $g(u, e) = (f(x_u, e), n_u + 1)$.

For any $U \subseteq X^+$, let $g(U, e) = \bigcup_{u \in U} g(u, e)$. This definition is extended to strings (rather than merely events) in the usual way. Also, define $g(s) = g(x_0^+, s)$ for brevity. Finally, we define the *augmented automaton* as the automaton $G^+ = (X^+, E, g, x_0^+)$ defined over augmented states that is induced from the original automaton G. Note that, since there is no bound on the count component of augmented states, G^+ might not be finite in size. (In Sect. IV, we will use a trimmed version of G^+ that avoids this problem). Figures 2.1 and 2.2 show an automaton and its corresponding augmented automaton.

Definition II.6 (Information State)**.** An information state (IS) is a subset $S \subseteq X^+$ of augmented states. We denote by $I = 2^{X^+}$ the set of information states. □

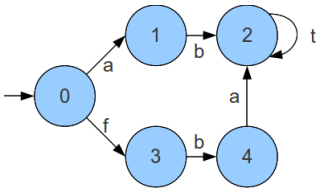


Figure 2.1: A finite state automaton with fault event f.

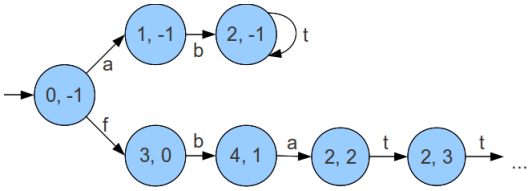


Figure 2.2: The augmented automaton for the automaton of Fig. 2.1

Definition II.7 (Information State Based Controller)**.** An information state based controller (or IS-controller) is a function $C : I \to 2^E$ that satisfies the two conditions of an admissible controller (i.e., $C(i) \supseteq E_o$ and $C(i) \cap (E_{uo} \cup E_f) = \emptyset$ for all $i \in I$). □

Henceforth, controllers will be assumed to be information state based unless stated otherwise.

With the information state defined, the next step is to construct a structure that encapsulates all admissible controllers (safe or not), which we call the total observer. This structure will serve as the basis from which we construct the most permissive observer, which will contain only the safe controllers. Constructing an observer for an automaton and a *fixed* set of unobservable events is a relatively simple task. To construct the observer when the set of observable events is a dynamic control decision, we must explicitly model the effect of the controller on the evolution of the information state.

Definition II.8 (Total Observer)**.** The total observer (TO) is defined as the bipartite automaton $TO = (Y \cup Z, \Gamma \cup E, h_{YZ} \cup h_{ZY}, y_0)$. Here, Y is the set of information states (i.e., $Y = I$) and Z is the set of information states augmented with control decisions (i.e., $Z = I \times \Gamma$). We use the notation $I(z)$ and $C(z)$ to denote z's information state

and control decision components, respectively, so that $z = (I(z), C(z))$. The function h_{YZ} defines transitions from Y states to Z states, which occur when a control decision is taken, and the function h_{ZY} defines transitions from Z states to Y states, which occur when a monitored event occurs (i.e., an observation). Thus, the alternation between control decisions and event observations (i.e, the run) also results in an alternation between Y and Z states. The initial state of TO is the Y state $y_0 = \{x_0^+\}$, the initial information state. The transition functions h_{YZ} and h_{ZY} are defined below. First, $z = h_{YZ}(y, \gamma)$ defined by the *unobservable reach* operation:

$$
\begin{aligned}
I(z) &= UR(y, \gamma) && (2.2)\\
&= \left\{ \begin{array}{l} v \in X^+ : (\exists u \in y)(\exists t \in (E \setminus \gamma)^*) \\ \quad \text{s.t. } v = g(u, t) \end{array} \right\} && (2.3)\\
&= \bigcup_{u \in y} \bigcup_{t \in (E \setminus \gamma)^*} g(u, t) && (2.4)\\
C(z) &= \gamma && (2.5)
\end{aligned}
$$

In words, this means that $I(z)$ is the set of augmented states reachable from some augmented state of the preceding Y state through some string of unmonitored events. Next, $y = h_{ZY}(z, e)$ for observation $e \in C(z)$ is given by:

$$
\begin{aligned}
y &= \{v \in X^+ : (\exists u \in I(z)) \text{ s.t. } v = g(u, e)\} && (2.6)\\
&= \bigcup_{u \in I(z)} g(u, e) && (2.7)
\end{aligned}
$$

In words, this means that y is the set of augmented states reachable through the *single* event e from some augmented state in the information state component of the preceding Z state. □

Because the total observer contains all admissible control decisions after each run and all possible event observances after each control decision, it contains every run

and, also, every admissible controller.

Definition II.9 (Y and Z State Controller Induced Information State Evolution)**.** Given a controller C, we define $IS_C^Y(y,s)$ to be the Y state that results from the occurrence of string s, when starting in Y state y. This can be computed as follows:

$$\begin{aligned} IS_C^Y(y,\varepsilon) &:= y \\ IS_C^Y(y,e) &:= \begin{cases} h_{ZY}(h_{YZ}(y,C(y)),e) & \text{if } e \in C(y) \\ y & \text{if } e \notin C(y) \end{cases} \\ IS_C^Y(y,es) &:= IS_C^Y(IS_C^Y(y,e),s) \end{aligned} \tag{2.8}$$

For brevity, we define $IS_C^Y(s) := IS_C^Y(y_0,s)$. Also define $IS_C^Z(z,s)$ analogously, with $IS_C^Z(s) := IS_C^Z(z_0,s)$ as before, where $z_0 = h_{YZ}(y_0,C(y_0))$ (which is well defined for a fixed controller). □

Definition II.10 (IS-Controller Induced Projection)**.** We redefine the controller induced projection $P_C(\rho,s)$ of Def. II.3 to $P_C(z,s)$ for IS-controllers:

$$\begin{aligned} P_C(z,\varepsilon) &:= \varepsilon \\ P_C(z,e) &:= \begin{cases} e & \text{if } e \in C(z) \\ \varepsilon & \text{if } e \notin C(z) \end{cases} \\ P_C(z,es) &:= \begin{cases} e.[P_C(z',s)] & \text{if } e \in C(z) \\ P_C(z,s) & \text{if } e \notin C(z) \end{cases} \\ &\text{where } z' = h_{YZ}(y',C(y')) \text{ and } y' = h_{ZY}(z,e) \end{aligned} \tag{2.9}$$

We define $P_C(s) := P_C(z_0,s)$ for brevity, as before. □

Lemma II.11 (Relation between projection and information state)**.** *For any string s and controller C, $I(IS_C^Z(s)) = \{v \in X^+ : \exists s' \text{ s.t. } P_C(s) = P_C(s') \wedge v = g(s')\}$.*

The proof of this lemma is given in Appendix 2. This lemma shows that the information state reached by controller C upon the occurrence of string s is the set

of augmented states reached through strings that cannot be distinguished from s by C.

2.5 The State Disambiguation Problem and the Most Permissive Observer

In this section, we show how to obtain the *most permissive observer* (MPO), the structure containing all safe controllers, from the total observer. We begin by briefly describing the state disambiguation problem and showing that the K diagnosability property can be formulated as a state disambiguation problem. This allows us to prove that the information state as defined is sufficient to uniquely determine the set of safe control decisions after any given run. We then proceed to define safety of Y and Z states and, finally, we define the MPO.

Definition II.12 (State Disambiguation Problem)**.** The state disambiguation problem is defined as a triple $\langle G^{sd}, \Sigma_o, T_{\text{spec}} \rangle$, where $G^{sd} = (X^{sd}, \Sigma, f^{sd}, x_0^{sd})$ is an automaton, $\Sigma_o \subseteq \Sigma$ is a set of monitorable events, and $T_{\text{spec}} \subseteq X^{sd} \times X^{sd}$ is a set of pairs that must not be confused. The state disambiguation problem consists of finding a controller C for G^{sd}, which chooses sensors to activate, such that the state of G^{sd} is never confused between any pair of states in the specification T_{spec}. The controller C is again defined as a function $C : R \to 2^{\Sigma_o}$ from runs to control decisions, as in Sect. 2.3. Using the notation defined in Sect. 2.3 of this work, we can define the problem formally as that of finding a controller C such that:

$$\begin{aligned} s_1, s_2 \in \mathcal{L}(G^{sd}) : P_C(s_1) = P_C(s_2) \\ \Rightarrow (f^{sd}(s_1), f^{sd}(s_2)) \notin T_{\text{spec}} \ . \end{aligned} \tag{2.10}$$

□

To formulate the K-diagnosability problem as a state disambiguation problem,

we specify each of G^{sd}, Σ_o, and T_{spec}:

- $G^{sd} = G_K^+$ which is the augmented automaton G^+, but restricted to augmented states having counts no greater than $K+1$.
- $\Sigma_o = E_o \cup E_s$ is the set of monitorable events (we assume that C is an admissible controller).
- Finally, we define T_{spec} as:

$$T_{\text{spec}} = \{(u,v) \in X^+ \times X^+ : N(u) = -1 \wedge N(v) = K+1\} \ . \tag{2.11}$$

Comparing Defs. II.4 and II.12, we see that this state disambiguation problem can be satisfied if and only if there do not exist two strings s_1 and s_2 with $P_C(s_1) = P_C(s_2)$, $N(g(s_1)) = -1$, and $N(g(s_2)) = K+1$. Recall that $N(g(s))$ is equal to -1 if there is no fault in s, and the number of events since a fault event otherwise. If we take s_1 and s_2 in this problem to correspond to s_N and s_Y of Def. II.4, we see that the two problems are identical, except for the fact that, in the definition of K-diagnosability s_Y must have *at least* $K+1$ events after a fault, whereas in T_{spec} the string s_2 has *exactly* $K+1$ events after a fault. To see that this makes no difference, suppose that there exist two strings s_N and s_Y that violate K-diagnosability and that s_Y has $r > K+1$ events after a fault. Then we may simply truncate s_Y to obtain an s'_Y with exactly $K+1$ events after a fault. If this shortens the projection $P_C(s_Y)$, we can truncate s_N as well to obtain an s'_N such that $P_C(s'_Y) = P_C(s'_N)$.

Definition II.13 (K-diagnosable binary function for information states)**.** An information state $i \in I$ violates K-diagnosability if there exist two augmented states $x_1^+, x_2^+ \in i$ where $x_1^+ = (x_1, -1)$ and $x_2^+ = (x_2, n)$ for some $n > K$. In light of the definition of T_{spec}, we define the K-diagnosability binary function for information states

$D_I : I \to \{0, 1\}$ as:

$$D_I(i) = \begin{cases} 0, & \exists u, v, \in I : (u, v) \in T_{\text{spec}} \\ 1, & \text{else} \end{cases} \tag{2.12}$$

In words, $D_I(i) = 1$ if and only if i does not violate the K-diagnosability property. □

Theorem II.14 (Formulation of the K-diagnosability property through the information state)**.** *Controller C is safe if and only if $D_I(I(z)) = 1$, for all reachable Z states. Mathematically:*

$$\begin{aligned} & \exists s \in \mathcal{L}(G) : z = IS_C^Z(s) \wedge \exists u, v \in I(z) \\ & \quad s.t.\ N(u) = -1 \ and \ N(v) = K + 1 \\ \Leftrightarrow \quad & \exists s_Y, s_N \in \mathcal{L}(G) : P_C(s_Y) = P_C(s_N), \\ & \quad N(g(s_N)) = -1 \ and \ N(g(s_Y)) = K + 1. \end{aligned}$$

Proof. ($\Leftarrow$) Since unobserved events cannot change the information state, we have that $IS_C^Z(s) = IS_C^Z(P_C(s))$. Thus, $P_C(s_Y) = P_C(s_N)$ implies $IS_C^Z(s_Y) = IS_C^Z(s_N) = z$. By definition, $g(s) \in I(IS_C^Z(s))$, for all s. Thus, $g(s_N) \in I(IS_C^Z(s_N)) = z$ and $g(s_Y) \in I(IS_C^Z(s_Y)) = z$ as well. We may therefore take $s = s_Y$, $u = g(s_N)$, and $v = g(s_Y)$.

($\Rightarrow$) Recall from Lem. II.11 that $I(IS_C^Z(s)) = \{v \in X^+ : \exists s' \text{ s.t. } P_C(s) = P_C(s') \wedge v = g(s')\}$. Then $u, v \in I(z)$ implies that there exists s_1, s_2 such that $P_C(s_1) = P_C(s_2) = P_C(s)$, $u = g(s_1)$, and $v = g(s_2)$. We simply take $s_N = s_1$ and $s_Y = s_2$. □

The above theorem has two consequences. First, because the safety of a controller is dependent on the Z information states that are reached and because the possible next Z states are determined by the current information state and control decision, it follows that there is no loss of generality by using IS-controllers instead of run-based

controllers. That is, any safe controller must make control decisions allowed by the MPO (although a run-based safe controller may make a different control decision from the same information state). The second consequence is that we have a test for determining whether or not a particular controller is safe, in terms of the Z states that are reachable. Specifically, they must all satisfy $D_I(I(z)) = 1$. Mathematically, we can write that controller C is safe if $D_I(I(IS_C^Z(s))) = 1$, for all $s \in \mathcal{L}(G)$. A minor additional consequence is that we can consider only Z states. This could have been guessed from the definition of the Z state as the unobservable reach of the preceding Y state, from which it follows that, for any y and any $C(y)$, $y \subseteq I(z)$ for $z = h_{YZ}(y, C(y))$.

The purpose of defining the MPO is to capture all controllers that satisfy the K-diagnosability property. In light of the preceding theorem and the fact that we can deterministically compute future Z states from the current Y or Z state and the sequence of future control decisions and observed events, we see that we can speak not only of safe controllers but also of safe information states. Specifically, we say that Y State y is safe if it currently satisfies the K diagnosability property and there exists some controller that maintains the K-diagnosability property for all future executions of the system. This is formalized in the following definition:

Definition II.15 (K-diagnosibility binary functions for Y and Z states)**.** Since we can choose control decisions but not event occurrences, we define two K-diagnosability binary functions, $D_Y : Y \to \{0, 1\}$ and $D_Z : Z \to \{0, 1\}$ (similar to D_I, but for Y and Z states) as follows:

$$D_Y(y) = \begin{cases} 1 & \text{if } D_I(y) = 1 \\ & \quad \text{and } \exists \gamma \in \Gamma : D_Z(h_{YZ}(y, \gamma)) = 1 \\ 0 & \text{else} \end{cases} \tag{2.13}$$

$$D_Z(z) = \begin{cases} 1 & \text{if } D_I(I(z)) = 1 \\ & \quad \text{and } D_Y(h_{ZY}(z,e)) = 1 \quad \forall e \in C(Z) \\ 0 & \text{else} \end{cases} \tag{2.14}$$

□

From these definitions, we can say that G is K-diagnosable if and only if $D_I(y_0) = 1$, and $\exists C(y_0) : D_I(h_{YZ}(y_0, C(y_0))) = 1$, and $\exists C(y_0) \forall e_0 \in C(y_0) : D_I(h_{ZY}(h_{YZ}(y_0, C(y_0)), e_0)) = 1$, and so forth. Put in terms of the information state evolution, we can equivalently say that G is K-diagnosable if and only if $\exists C$ such that $\forall s \in \mathcal{L}(G)$, $D_I(IS_C^Y(s)) = 1$ and $D_I(I(IS_C^Z(s))) = 1$ (in practice, the second condition is sufficient). Thus the alternation of existential and universal quantifiers in Def. II.15 implicitly captures the idea that there must exist some controller such that K-diagnosability holds for all possible strings of events. This is the same conclusion that is reached from Thm. II.14. Note that the above definitions are akin to the controllable predecessor of game theory, with the information states such that $D_I(\cdot) = 1$ playing the role of the base set of winning states, as is done in *Cassez and Tripakis* (2008). We can now use the above definition to define the safety of a control decision.

Definition II.16 (Safe Control Decision)**.** Control decision γ is safe from Y state y if and only if $D_Z(h_{YZ}(y, \gamma)) = 1$, since we know that there exists a future sequence of safe control decisions in this case. □

The above definition has the consequence that after any run ρ resulting in Y state y, the set of safe control decisions is uniquely determined by the information state y. Thus, there is no loss of generality in using information state based controllers.

Definition II.17 (Fault diagnosis binary function)**.** Define the fault diagnosis binary

function $D_F : Y \to \{0, 1\}$ as follows:

$$D_F(y) = \begin{cases} 1 & \text{if } N(u) \neq -1, \quad \forall u \in y \\ 0 & \text{else} \end{cases} \tag{2.15}$$

In words, this means that $D_F(y) = 1$ if and only if all possible executions $s \in \mathcal{L}(G)$ resulting in information state y had a fault occurrence. A Y state y satisfying $D_F(y) = 1$ is called *diagnosed*. □

Definition II.18 (Structural Definition of the Most Permissive Observer)**.** The most permissive observer is defined as the *trim* of the total observer when removing Y states such that $D_Y(\cdot) = 0$ and Z states such that $D_Z(\cdot) = 0$. The trim operation removes these states and all transitions involving them, as well as any states and transitions that become unreachable from the initial state. To make this automaton finite, we also replace any Y state such that $D_F(y) = 1$ by a "fault detected" state F, and make this state terminal in the sense that there are no transitions out of it (i.e., we make no further control decisions from F). □

Theorem II.19 (Correctness of the MPO)**.** *If $K < \infty$, then a controller C is safe if and only if it only makes control decisions allowed by the MPO.*

Proof. The contrapositive of the "only if" part of this assertion follows immediately from Thm. II.14. To see the "if" part of the assertion, consider any string $s_Y \in (E \setminus E_f)^* E_f E^{K+1}$. Since new control decisions are made only upon event observances, at most $K+1$ control decisions will be made after the occurence of a fault event before reaching a Z information state containing an augmented state with a count of $K+1$. If all the control decisions are safe, then this information state cannot also contain an augmented state with a count of -1, and hence the fault will be diagnosed. Thus, if K is finite, a controller which makes only safe control decisions must reach the fault diagnosed state within $K + 1$ events of a fault event occurring. From Lem.

II.11, it follows that there does not exist any $s_N \in (E \setminus E_f)^* \cap \mathcal{L}(G)$ such that $P_C(s_N) = P_C(s_Y)$, which proves that controller C is safe. $\square$

Thus, the MPO as defined does indeed solve Problem II.5 when K is finite. This result was also proven in *Cassez and Tripakis* (2008) for their construction of the MPO. If K is infinite, then infinite sequences of control decisions without a definitive diagnosis after a fault event are possible. See *Thorsley and Teneketzis* (2007) for an example of this. An example of the MPO is useful at this point.

Example II.20 (A simple example)**.** Figures 2.3 and 2.4 show the total and most permissive observer for the automaton of Fig. 2.1. Note that the TO shown does not have any transitions out of Y and Z states such that $D_I(\cdot) = 0$ for all finite K and shows only states reachable from y_0. We have also not shown any transitions out of diagnosed Y states or from Z states for events that would lead to an empty information state (e.g., there is no b transition out of Z state z_3). Finally, we omit (in both the TO and MPO) any further control decisions from any Y state from which it is clear that no fault has occurred in the past and none can occur in the future (such as y_1).

If we take any $K \geq 2$, then the removal of all Y and Z states such that $D_I(\cdot) = 0$ results in the removal of all states of the TO marked with a "X", which leaves state y_2 of the TO with no safe control decisions. If we take $K = 1$, then $D_I(y_2) = 0$ as well. In either case, y_2 is removed, leading to the removal of z_2. Intuitively, it is initially necessary to choose to monitor event b for otherwise it will not be possible to differentiate between the string $fbat^n$ and the string a, which means K-diagnosability is violated for any K. Thus, it is indeed possible for a Y or Z state to satisfy $D_I(\cdot) = 1$ but still not be safe. It can be verified that $D_Y(\cdot) = 1$ for all remaining Y states and that $D_Z(\cdot) = 1$ for all remaining Z states. To verify that this is intuitively correct, consider what happens after choosing to initially monitor only b and then observing event b. At this point, it is now necessary to monitor event a, for otherwise it will not

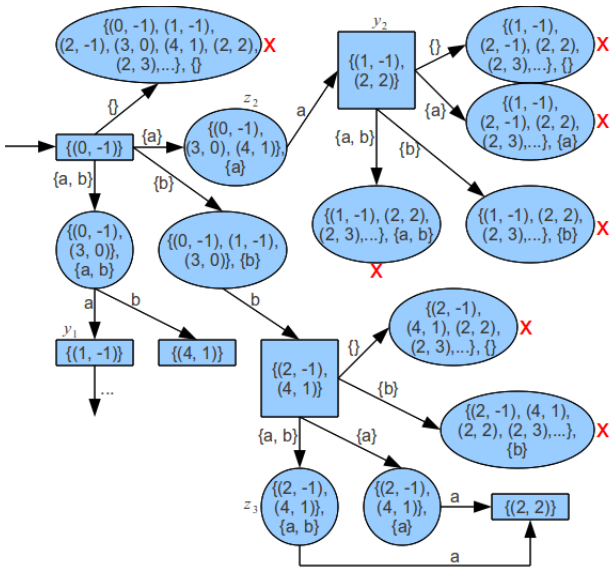


Figure 2.3: The total observer for the automaton of Fig. 2.1, with events classified as follows: $E_o = \emptyset$, $E_s = \{a, b\}$, $E_{uo} = \{t\}$, and $E_f = \{f\}$.

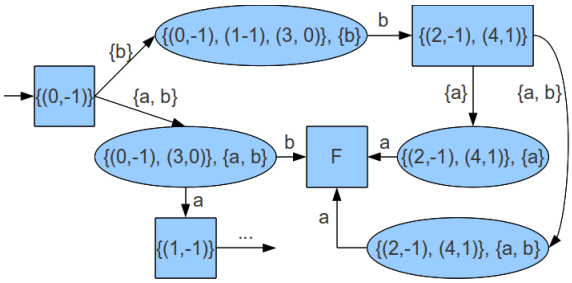


Figure 2.4: The MPO for the automaton of Fig. 2.1, for any $K \geq 1$. We used the convention of *Cassez and Tripakis* (2008) by marking Y states with squares and Z states with circles.

be possible to differentiate between the string ab and the string $fbat^n$, which again violates K-diagnosability for any K. Notice that we can achieve 0-diagnosability by choosing to monitor $\{a, b\}$ initially, and only 1-diagnosability if we choose to monitor only $\{b\}$. □

With the structure of the MPO defined, it remains to describe an efficient method of constructing it. One possibility is to use the method described in *Cassez and Tripakis* (2008), in which the most permissive observer is obtained as the solution of a two player safety game where player 1 must prevent the system from reaching the

unsafe set of states. The same method can be applied here, where the analogous set of "unsafe states" to be avoided consists of those Y and Z states such that $D_I(\cdot) = 0$. However, we will not use this method as a means of computing the most permissive observer in this work. We will instead use the extended specification of state disambiguation problems, which will be described in the next section.

2.6 The Extended Specification and Properties of the MPO

In this section, we present the extended specification and explain how it allows for a simple test to determine the safety of the Y and Z states of the MPO.

Definition II.21 (Extended Specification)**.** We repeat the definition of the extended specification found in *Wang et al.* (2009):

$$T^e_{\text{spec}} = \left\{ \begin{array}{l} (u,v) \in X^+ \times X^+ : \exists s_1, s_2 \text{ s.t. } P(s_1) = P(s_2) \\ \quad \text{and } (g(u,s_1), g(v,s_2)) \in T_{\text{spec}} \end{array} \right\} \tag{2.16}$$

In words, the extended specification is defined as the set of all augmented state pairs that must not be confused because even if all the sensors in $E_o \cup E_s$ are turned on for the rest of time, there still exists some sequence of events such that some pair in T_{spec} will be confused. □

Definition II.22 (Extended specification binary function for information states)**.** In light of the definition of T^e_{spec}, we define the extended specification binary function for information states $D^e_I : I \to \{0,1\}$ as follows:

$$D^e_I(i) = \left\{ \begin{array}{ll} 0, & \exists u, v, \in I : (u,v) \in T^e_{\text{spec}} \\ 1, & \text{else} \end{array} \right. \tag{2.17}$$

In words, $D^e_I(i) = 1$ if and only if i does not violate the extended specification. □

The following theorems show how the extended specification is useful.

Lemma II.23 (Equality of projection when everything is observed is equivalent to equality of projection under all controllers)**.** *For any two strings $s_1, s_2 \in E^*$ and starting Z state z, $P(s_1) = P(s_2) \Leftrightarrow \forall C, P_C(z, s_1) = P_C(z, s_2)$.*

Proof. ($\Leftarrow$) Clearly, if $P_C(z, s_1) = P(z, s_2)\ \forall C$, then it is true for the particular controller C_{all} defined by $C_{\text{all}}(y) = E_o \cup E_s,\ \forall y \in I$. But this is exactly the same as taking the natural projection P.
($\Rightarrow$) Certainly, events that can never be observed make no difference in the projection that is computed by some controller C. Thus, we may write $P_C(z, s_1) = P_C(z, P(s_1))$ and $P_C(z, s_2) = P_C(z, P(s_2))$, from which it follows that $P(s_1) = P(s_2) \Rightarrow \forall C, P_C(z, s_1) = P_C(z, s_2)$. □

Theorem II.24 (Safety is equivalent to satisfying the extended specification for Z states)**.** *For any Z state z, we have $D_Z(z) = D_I^e(I(z))$. That is, a Z state is safe if and only if it satisfies the extended specification.*

Proof. Define $\mathcal{L}(G, S(I(z))) := \bigcup_{x \in S(I(z))} \mathcal{L}(G, x)$, the set of strings that can occur

after reaching Z state z. Then:

$$
\begin{array}{lll}
 & D_Z(z) = 0 & \\
\Leftrightarrow & \begin{array}{l}\forall C, \exists s \in \mathcal{L}(G, S(I(z))) : \\ \quad D_I(I(IS_C^Z(z,s))) = 0\end{array} & \text{Thm. II.14} \\
 & \forall C, \exists s \in \mathcal{L}(G, S(I(z))), & \\
\Leftrightarrow & \quad \exists v_1, v_2 \in I(IS_C^Z(z,s)) : & \text{Def. II.13} \\
 & \quad (v_1, v_2) \in T_{\text{spec}} & \\
 & \forall C, \exists s_1, s_2 \in \mathcal{L}(G, S(I(z))), & \\
 & \quad \exists u_1, u_2 \in I(z) : & \\
\Leftrightarrow & \quad P_C(z, s_1) = P_C(z, s_2), & \\
 & \quad \text{and } (g(u_1, s_1), g(u_2, s_2)) \in T_{\text{spec}} & \\
 & \exists s_1, s_2 \in \mathcal{L}(G, S(I(z))), & \\
\Leftrightarrow & \quad \exists u_1, u_2 \in I(z) : P(s_1) = P(s_2) & \text{Lemma II.23} \\
 & \quad \text{and } (g(u_1, s_1), g(u_2, s_2)) \in T_{\text{spec}} & \\
\Leftrightarrow & \exists u_1, u_2 \in I(z) : (u_1, u_2) \in T_{\text{spec}}^e & \text{Def. II.21} \\
\Leftrightarrow & D_I^e(I(z)) = 0 & \text{Def. II.22}
\end{array}
$$

The fourth equivalence follows in the forward direction by taking the particular controller C_{all}. In the backward direction, we have a statement of the form $\exists(\cdot) : P(s_1) = P(s_2) \ldots$ and invoke Lemma II.23 to obtain a statement of the form $\exists(\cdot), \forall C : P_C(z, s_1) = P_C(z, s_2) \ldots$, which implies the previous statement of the form $\forall C, \exists(\cdot) : P_C(z, s_1) = P_C(z, s_2) \ldots$, since $\exists\forall(\cdot)$ is stronger than $\forall\exists(\cdot)$. □

Note that Thm. II.24 immediately implies that the safety of a Z state is solely dependent on its information state component, and not on its associated control decision.

A few corollaries follow from this result.

Corollary II.25 (Monotonicity Properties)**.**

(i) If Z state z_1 is safe then so is any Z state z_2 satisfying $I(z_2) \subseteq I(z_1)$.

(ii) Any control decision γ that is safe in Y state y_1 is also safe in Y state $y_2 \subseteq y_1$.

(iii) If control decision γ_1 is safe in Y state y, then so is any control decision $\gamma_2 \supseteq \gamma_1$.

Proof.

(i) Immediate from Thm. II.24 and Def. II.22, since $I(z_2) \subseteq I(z_1) \Rightarrow D_I^e(I(z_2)) \geq D_I^e(I(z_1))$.

(ii) Immediate from part (i), since $y_2 \subseteq y_1$ implies $I(h_{YZ}(y_2, \gamma)) \subseteq I(h_{YZ}(y_1, \gamma))$.

(iii) Immediate from part (i), since $\gamma_2 \supseteq \gamma_1$ implies $I(h_{YZ}(y, \gamma_2)) \subseteq I(h_{YZ}(y, \gamma_1))$. □

Theorem II.26 (Safety is equivalent to satisfying the extended specification for Y states)**.** *For any Y state y, we have $D_Y(y) = D_I^e(y)$. That is, a Y state is safe if and only if it satisfies the extended specification.*

Proof.

$$
\begin{aligned}
& D_Y(y) = 0 && \\
\Leftrightarrow\ & D_I(y) = 0 \\
& \quad \text{or } \forall \gamma : D_Z(h_{YZ}(y,\gamma)) = 0 && \text{Def. II.15} \\
\Leftrightarrow\ & D_I(y) = 0 \\
& \quad \text{or } D_Z(h_{YZ}(y, E_o \cup E_s)) = 0 && \text{Cor. II.25 (iii)} \\
\Leftrightarrow\ & D_I(y) = 0 \\
& \quad \text{or } D_I^e(I(h_{YZ}(y, E_o \cup E_s))) = 0 && \text{Thm. II.24} \\
& D_I(y) = 0 \\
\Leftrightarrow\ & \quad \text{or } \exists v_1, v_2 \in I(h_{YZ}(y, E_o \cup E_s)) : && \text{Def. II.21} \\
& \quad (v_1, v_2) \in T^e_{\text{spec}} \\
& D_I(y) = 0 \\
& \quad \text{or } \exists u_1, u_2 \in y, \\
\Leftrightarrow\ & \quad \exists s_1, s_2 \in (E_{uo} \cup E_f)^* : \\
& \quad (g(u_1, s_1), g(u_2, s_2)) \in T^e_{\text{spec}} \\
& D_I(y) = 0 \\
\Leftrightarrow\ & \quad \text{or } \exists u_1, u_2 \in y : (u_1, u_2) \in T^e_{\text{spec}} && \text{Def. II.21} \\
\Leftrightarrow\ & D_I^e(y) = 0 && \text{Def. II.22}
\end{aligned}
$$

The fifth equivalence follows from the definition of h_{YZ} as the unobservable reach. □

Theorems II.24 and II.26 have important implications. By making use of the extended specification, we can determine the safety of any control decision $\gamma \in \Gamma$, from any Y state y by checking if the resulting Z state satisfies the extended specification. That is, γ is a safe control decision from y if and only if $D_I^e(I(h_{YZ}(y, C(y))) = D_I^e(UR(y, C(y))) = 1$. This can be done in $O(K|X||E| + K|X|^2)$ time, since computing the unobservable reach can be done through a $O(K|X||E|)$ time depth-first search over G_K^+ and evaluating $D_I^e(\cdot)$ can be done in $O(K|X|^2)$ time. Thus, we can determine safe control decisions from any Y state y without constructing any part of the TO

beyond the immediate successors of y. This implies that the MPO can be constructed with a search algorithm that need only visit safe states of the TO (i.e., those that will remain in the MPO) and their immediate successors. This results in savings in both running time and memory and would not be the case if we were to use the algorithm presented in *Cassez and Tripakis* (2008). Another important consequence of the use of the extended specification is that, if $|E_s|$ is small, we can compute an "on the fly" minimal solution (i.e., a safe controller C such that there does not exist any other safe controller $C' \neq C$ such that $C'(y) \subseteq C(y)$ for all $y \in Y$ satisfying $D_Y(y) = 1$). Clearly, any controller that makes minimal safe control decisions at each Y state is a minimal controller (the converse is not true). Thus, it suffices to find a minimal safe control decision at each Y state to construct a minimal controller on the fly, and this requires time exponential in E_s, but only quadratic in $|X|$, at each new Y state.

Example II.27 (A simple example revisited)**.** To illustrate the use of the extended specification, let us now revisit Ex. II.20. From the definition of T_{spec}, we know that $((1,-1),(2,K+1)) \in T_{\text{spec}}$. The two strings $s_1 = a$ and $s_2 = at^{K-1}$ satisfy $P(s_1) = P(s_2)$, $g((0,-1),s_1) = (1,-1)$, and $g((4,1),s_2) = (2,K+1)$. Thus, we have $((0,-1),(4,1)) \in T^e_{\text{spec}}$, for any finite $K > 0$. If $K = 0$, then $((0,-1),(4,1)) \in T_{\text{spec}} \subseteq T^e_{\text{spec}}$. Thus, for any finite K, $((0,-1),(4,1)) \in T^e_{\text{spec}}$, from which it follows that state z_2 of Fig. 2.4 does not satisfy the extended specification and will not be in the MPO. The use of the extended specification therefore allows us to construct the MPO without exploring y_2 and its four successors. □

2.7 Reducing the Information State

In this section, we prove new results using a monotonicity property on the extended specification. Specifically, we use this to show that we can "reduce" our information state. That is, we will show that, as currently defined, our information

state carries more information than what is strictly necessary for the problem of K-diagnosability. In order to simplify notation in what follows, we write that $v' >^+ v$ for augmented states v and v' if $S(v') = S(v)$ and $N(v') > N(v)$. Let $<^+$, $\geq^+$, and $\leq^+$ be similarly defined for augmented states.

Theorem II.28 (Monotonicity Property for the Extended Specification)**.** *Suppose that $(u, v) \in T^e_{spec}$. Then, for any $v' >^+ v$, we also have $(u, v') \in T^e_{spec}$.*

Proof. Suppose that $(u, v) \in T^e_{\text{spec}}$. Then:

$$\exists s_1, s_2 : P(s_1) = P(s_2), \quad (g(u, s_1), g(v, s_2)) \in T_{\text{spec}} \ .$$

We know that $N(g(u, s_1)) = -1$ and $N(g(v, s_2)) = K + 1$. Clearly, $v' >^+ v \Rightarrow g(v', s_2) >^+ g(v, s_2)$. Truncate s_2 to obtain s_2' such that $N(g(v', s_2)) = K + 1$. Similarly, truncate s_1 to obtain s_1' such that $P(s_1') = P(s_2')$. Then s_1' and s_2' satisfy the three conditions that $P(s_1') = P(s_2')$, $N(g(u, s_1')) = -1$, and $N(g(v', s_2')) = K+1$, which implies that $(u, v') \in T^e_{\text{spec}}$. □

The above theorem allows us to reduce the extended specification to a $|X| \times |X|$ table filled with elements from $\{-1, 0, \ldots, K+1\}$, where an entry of n at location (x_1, x_2) signifies that $((x_1, -1), (x_2, n)) \in T^e_{\text{spec}}$ and that, for all $n' < n$, $((x_1, -1), (x_2, n')) \notin T^e_{\text{spec}}$. We will see with the next definition and theorems that this has even more significant consequences.

Definition II.29 (Information State Reducing Function)**.** Define $R : I \to I$ by:

$$R(i) = \{u \in i : [N(u) = -1] \vee [\nexists u' \in i \text{ s.t. } u' >^+ u]\} \ . \tag{2.18}$$

Also define the notation $m(i) = \{u \in i : N(u) = -1\}$ and $M(i) = \{u \in i : \nexists u' \in i \text{ s.t. } u' >^+ u\}$, so that $R(i) = m(i) \cup M(i)$. Define $R : Z \to Z$ by

$R(z) = (R(I(z)), C(z))$. Finally, define reduced versions of h_{YZ}, UR, h_{ZY}:

$$h^R_{YZ}(y, C(y)) = R(h_{YZ}(y, C(y))) \tag{2.19}$$

$$RUR(y, C(y)) = R(UR(y, C(y))) \tag{2.20}$$

$$h^R_{ZY}(z, e) = R(h_{ZY}(z, e)) \tag{2.21}$$

□

In words, $R(\cdot)$ *reduces* an information state by keeping only those augmented states within it that have a count of -1 and, for each state in X, keeping only the augmented state with that state component that has the highest count. As the following corollary will show, this information is sufficient to determine the safety of a Y or Z state.

Corollary II.30 (Reduced information state carries all necessary information in determining safety)**.** *For any information state $i \in I$, $D^e_I(i) = D^e_i(R(i))$.*

Proof. Obviously, $R(i) \subseteq i$, so that $D^e_I(R(i)) \geq D^e_I(i), \quad \forall i \in I$. Now suppose to the contrary that for some $i \in I$, $D^e_I(i) \neq D^e_i(R(i))$. Then it must be that $D^e_i(R(i)) = 1$ and $D^e_I(i) = 0$. Thus, $\exists u, v \in i : (u, v) \in T^e_{\text{spec}}$. Since $N(u) = -1$, we know that $u \in R(i)$ as well. Then it must be that $v \notin R(i)$. Hence $N(v) \neq -1$ and, since $v \in i$ but $v \notin R(i)$, there exists some $v' \in R(i)$ such that $v' >^+ v$. But by Thm. II.28 and $(u, v) \in T^e_{\text{spec}}$, we know that $(u, v') \in T^e_{\text{spec}}$, so that $D^e_i(R(i)) = 0$ also, a contradiction. □

This proves that $R(\cdot)$ conserves all the necessary information for determining the safety of a Y or Z state. However, it is still possible that this "filtering out" of information in the present could change safety properties of Y or Z states in the future (i.e., those Y and Z states that are reachable further in the execution of the system). The following two theorems preclude this possibility.

Theorem II.31 (There is no loss in applying h^R_{ZY} to a reduced Z state)**.** *For any Z state z and event $e \in C(z)$, $h^R_{ZY}(z,e) = h^R_{ZY}(R(z),e)$.*

Proof. We seek to prove that $m(h_{ZY}(z,e)) = m(h_{ZY}(R(z),e)))$ and that $M(h_{ZY}(z,e)) = M(h_{ZY}(R(z),e)))$. From the definition of h_{ZY}, $\{v \in h_{ZY}(z,e)\} = \bigcup_{u \in I(z)} g(u,e)$. Since $e \notin E_f$,

$$\begin{aligned} &m(h_{ZY}(z,e)) = \textstyle\bigcup_{u \in m(I(z))} g(u,e) \\ &= \textstyle\bigcup_{u \in m(I(R(z)))} g(u,e) = m(h_{ZY}(R(z),e)). \end{aligned}$$

Next, we claim that, for any $i_1, i_2 \in I$:

$$M(i_1) \subseteq i_2 \wedge M(i_2) \subseteq i_1 \Rightarrow M(i_1) = M(i_2) \tag{2.22}$$

To see this, consider any $v \in M(i_1) \subseteq i_2$ and suppose that $v \notin M(i_2)$. Then there exists some $v' \in M(i_2)$ such that $v' >^+ v$. But $M(i_2) \subseteq i_1$, so this would imply $v' \in i_1$, contradicting $v \in M(i_1)$. Thus, $M(i_1) \subseteq M(i_2)$. We can prove $M(i_2) \subseteq M(i_1)$ by a symmetrical argument. Next, we claim that:

$$\begin{aligned} &M(i_1) = M(i_2), \\ &\quad \text{where } i_1 = \textstyle\bigcup_{u \in I(z)} g(u,e) \text{ and } i_2 = \bigcup_{u \in M(I(z))} g(u,e) \end{aligned} \tag{2.23}$$

From the definition of M, it follows that $M(i_2) \subseteq i_2 \subseteq i_1$. By Eq. (2.22), it therefore suffices to prove that $M(i_1) \subseteq i_2$. Consider any $v \in M(i_1)$ and let $u \in I(z)$ be such that $g(u,e) = v$. Suppose that $u \notin M(I(z))$. Then there exists some $u' \in I(z)$ such that $u' >^+ u$. This in turn implies that $g(u',e) >^+ g(u,e) = v$, which contradicts $v \in M(i_1)$. It follows that $u \in M(I(z))$ and hence that $v = g(u,e) \in i_2$, which proves

that $M(i_1) \subseteq i_2$. Finally, we obtain:

$$\begin{aligned}
& M(h_{ZY}(z,e)) \\
= \; & M\left(\bigcup_{u\in I(z)} g(u,e)\right) && \text{Def. of } h_{ZY} \\
= \; & M\left(\bigcup_{u\in M(I(z))} g(u,e)\right) && \text{Eq. (2.23)} \\
= \; & M\left(\bigcup_{u\in M(I(R(z)))} g(u,e)\right) && \text{Def. of } M(\cdot) \\
= \; & M\left(\bigcup_{u\in I(R(z))} g(u,e)\right) && \text{Eq. (2.23)} \\
= \; & M(h_{ZY}(R(z),e) && \text{Def. of } h_{ZY}
\end{aligned}$$

which completes the proof. □

Theorem II.32 (There is no loss in applying h^R_{YZ} to a reduced Y state)**.** *For any Y state y and control decision $C(y)$, $h^R_{YZ}(y, C(y)) = h^R_{YZ}(R(y), C(y))$.*

Proof. Similar to the previous proof, except that we consider unobservable strings of events rather than single events. □

By induction, Thms. II.31 and II.32 along with Cor. II.30 show that we do not lose any information relevant to determining safety of Y or Z states by working solely with reduced information states. Significantly, this substantially reduces the number of "distinct" information states. Without this reduction, the number of information states that may be present in the MPO is $2^{(K+2)|X|}$, since there are $|\{-1, 0, 1, \ldots, K\}| = K + 2$ values for the count component of each augmented state in the information state. For reduced information states ri, we must indicate, for each $x \in X$, whether or not $(x, -1) \in ri$ and what the maximal value of n is such that $(x, n) \in ri$. If we use -1 to denote both the case where this maximal value is -1 and the case when $x \notin S(ri)$ (where we determine which case it is by checking if $(x, -1) \in ri$), then we obtain only $2^{|X|} \cdot (K+2)^{|X|}$ distinct reduced information states that may be present in the MPO. *This reduces the space complexity of the MPO from exponential in K to polynomial in K.*

Following is a second MPO example that demonstrates the usefulness of the reduced information state.

Example II.33 (MPO with and without reduced information state)**.**

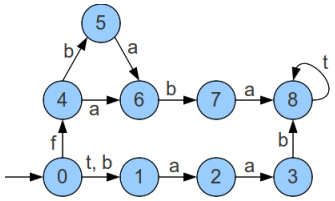


Figure 2.5: A finite state automaton. Events are classified as follows: $E_o = \emptyset$, $E_s = \{a, b\}$, $E_{uo} = \{t\}$, and $E_f = \{f\}$.

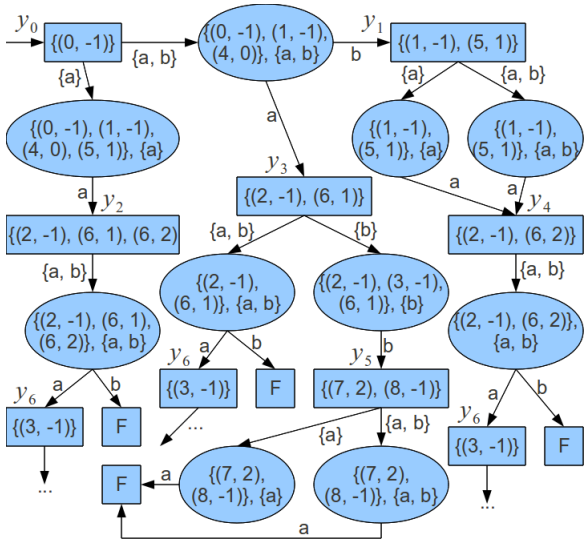


Figure 2.6: The MPO corresponding to the automaton of Fig. 2.5, with $K = 2$, not using reduced information states.

Consider the automaton of Fig. 2.5. The entire extended specification has $|X|^2 = 81$ "critical values", the minimum value of n such that $((x_1, -1), (x_2, n)) \in T^e_{\text{spec}}$, for each $x_1, x_2 \in X$. Of these, only four actually restrict behavior: $((0, -1), (6, 0))$, $((3, -1), (6, 2))$, $((2, -1), (7, 0))$, and $((8, -1), (8, 0))$. These values can be determined by inspection for this example, by computing longest strings with the same projection for each pair (see Appendix 1 for details on this method for computing T^e_{spec}). For $((0, -1), (6, 0))$, take $s_1 = ba$ and $s_2 = bat^\infty$. For $((3, -1), (6, 2))$, take $s_1 = b$ and

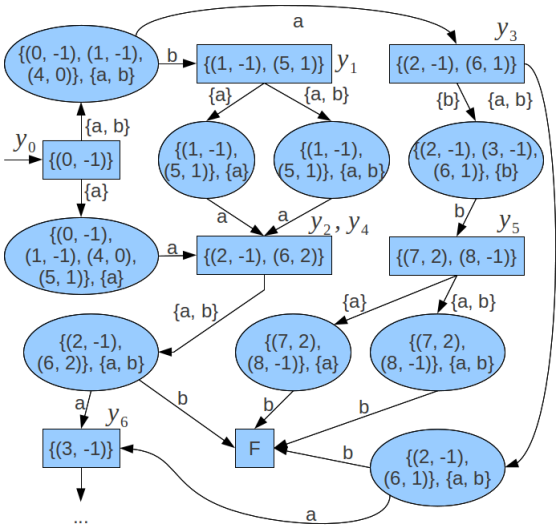


Figure 2.7: The MPO corresponding to the automaton of Fig. 2.5, with $K = 2$, using reduced information states.

$s_2 = b$. For $((2, -1), (7, 0))$, take $s_1 = a$ and $s_2 = at^\infty$. Finally, for $((8, -1), (8, 0))$, take $s_1 = \varepsilon$ and $s_2 = t^\infty$. The remaining critical values are either irrelevant because the corresponding pair of states does not occur, or are relevant but do not require any more events to be monitored. The MPOs without and with reduced information states are shown in figures 2.6 and 2.7, respectively. If we choose not to monitor event a initially, the unobservable reach will include both augmented states $(0, -1)$ and $(6, 1)$, which causes a lack of K-diagnosability since $((0, -1), (6, n)) \in T^e_{\text{spec}}$ for any $n \geq 0$. From Y states y_1, y_2, and y_4, we must monitor event a since the unobservable reach from any of these Y states would otherwise include both augmented states $(3, -1)$ and $(6, 2)$, and $((3, -1), (6, n)) \in T^e_{\text{spec}}$ for any $n \geq 2$. From Y states y_2, y_3, and y_4, we must monitor event b since the unobservable reach from any of these Y states would otherwise include both augmented states $(2, -1)$ and one of $(7, 2)$ or $(7, 3)$, and $((2, -1), (7, n)) \in T^e_{\text{spec}}$ for any $n \geq 0$. Finally, it is also necessary to monitor event a from Y state y_5, since the unobservable reach would otherwise include both augmented states $(8, -1)$ and $(8, 3)$, and $((8, -1), (8, n)) \in T^e_{\text{spec}}$ for any $n \geq 0$. As in the MPO of Ex. II.20, we omitted the portion of the MPO after which we can

determine that no fault has occurred in the past and none can occur in the future.

We can confirm the validity of this MPO by observing that the faulty and non-faulty languages are $f(\varepsilon + b)abat^*$ and $(\varepsilon + b)aabt^*$, respectively, whereas the faulty and non-faulty languages of Ex. II.20 are $fbat^*$ and abt^*, respectively. Thus, once we have observed a first occurrence of event a, the structure of the MPO from that point on should be similar to the structure of the MPO in Ex. II.20. Indeed, upon observing event a we find ourselves in one of Y states y_2, y_3, or y_4. If we have potentially reached an augmented state with a count of 2 (i.e., in y_2 or y_4), then the MPO from these states onwards has the same structure as the MPO of Ex. II.20, but with $K = 0$. If, on the other hand, at most one event has occurred after a fault (i.e., in y_3), then the MPO from this state onwards has the same structure as the MPO of Ex. II.20 (i.e., with $K = 1$).

Notice that MPO states y_2 and y_4 have the same reduced versions (namely $(2, -1), (6, 2)$) and the same structure from that point on. On the other hand, MPO state y_3 is also very similar to y_2 and y_4, but has a different reduced version (namely $(2, -1), (6, 1)$) and therefore a different structure from that point on. □

2.8 Constructing the MPO

In this section, we provide an algorithm for constructing the MPO and determine its running time. The algorithm assumes that T^e_{spec} has already been computed. An algorithm for computing T^e_{spec} and its running time are given in Appendix 1. The algorithm also makes use of a specialized algorithm for computing the reduced unobservable reach, which has a better running time than simply computing the unobservable reach and reducing the resulting information state. This algorithm and its running time are also given in Appendix 1.

The basic outline of an implemented algorithm for constructing the MPO is shown in Algorithm 1. The algorithm is based on a depth-first search (DFS). The parameter

G represents the finite-state automaton, the parameter y is a Y state, the parameter $Tespec$ is the extended specification, the parameter Γ is the set of admissible control decisions, and the parameter MPO represents the MPO automaton, with $MPO.YZ$ being its set of states and $MPO.h$ being its transition function. Before calling MakeMPO, it is necessary to check if G is diagnosable. If so, then the initial call will be MakeMPO(G, y_0, $Tespec$, Γ, MPO), with $MPO.YZ$ initialized to $\{y_0\}$ and $MPO.h$ initialized to the empty set.

Algorithm 1 searches through the space of Y states and, for each encountered Y state y, checks if $D_F(y) = 1$ (lines 2-4) and finds the safe control decisions (line 7). For each safe control decision, a transition is added from y to the next z state (line 11). If a control decision leads to a Z state z that is already in the MPO, the algorithm moves on to another control decision (lines 12-14). If z is not already in the MPO, it is added to the MPO (line 15), and the algorithm enters the inner for loop. The inner for loop considers each possible observation $e \in \gamma$ which could occur from z and computes the next Y state y'. If y' is an empty information state (which occurs if e cannot be the next observation from z), the algorithm moves on to another event $e \in \gamma$ (lines 18-20). Otherwise, a transition is added from z to y' (line 21). If y' is non-empty and not already in the MPO, it is added to the MPO (line 23), and the algorithm recurses on y' (line 24). Since there is a finite number of augmented states with count at most $K + 1$, there is a finite number of information states that will be traversed and the algorithm must eventually terminate.

Theorem II.34 (Correctness of MakeMPO)**.** *Algorithm 1 correctly constructs the MPO.*

Proof. Def. II.18 defines the MPO as the accessible part of the TO after removing the Y states such that $D_Y(\cdot) = 0$ and the Z states such that $D_Z(\cdot) = 0$. It follows that the MPO can be correctly constructed by a DFS over the TO which visits all the remaining accessible Y and Z states. By Thm II.24, $D_Z(z) = D_I^e(I(z))$. By Def.

Algorithm 1: Algorithm for constructing the MPO

```
1: procedure MakeMPO(G, y, Tespec, Γ, MPO)
2:     if DF(y) = true then                          ▷ Fault can be diagnosed
3:         return
4:     end if
5:     for all γ ∈ Γ do                    ▷ Try all admissible control decisions
6:         ur ← GetRUR(y, γ)    ▷ Get reduced unobservable reach for next z state
7:         if DeI(ur, Tespec) = false then                ▷ γ is not safe from y
8:             continue              ▷ Try the next admissible control decision
9:         end if
10:        z ← (ur, γ)
11:        MPO.h ← MPO.h ∪ {(y, γ, z)}                          ▷ z = h(y, γ)
12:        if z ∈ MPO.YZ then                        ▷ z is already in MPO.YZ
13:            continue          ▷ Try the next admissible control decision
14:        end if
15:        MPO.YZ ← MPO.YZ ∪ {z}                          ▷ Add z to MPO.YZ
16:        for all e ∈ γ do                                    ▷ Try all events
17:            y' ← Next(ur, e)                    ▷ Get next reduced y state
18:            if y' = ∅ then                                    ▷ y' is empty
19:                continue                                ▷ Try the next event
20:            end if
21:            MPO.h ← MPO.h ∪ {(z, e, y')}                       ▷ y' = h(z, e)
22:            if y' ∉ MPO.YZ then              ▷ y' is not already in MPO.YZ
23:                MPO.YZ ← MPO.YZ ∪ {y'}                  ▷ Add y' to MPO.YZ
24:                MakeMPO(G, y', Tespec, sl, E, MPO)
25:            end if
26:        end for
27:    end for
28: end procedure
```

II.15, $D_Z(z) = 1$ implies $D_Y(y) = 1$, for all successor y states. Thus it suffices for the DFS to visit all encountered Y and Z states other than the Z states such that $D_I^e(I(z)) = 0$, which is precisely what Algorithm 1 does. □

Proposition II.35 (Running time of MakeMPO)**.** *The running time of Algorithm 1 is in* $O([(2(K+2))^{|X|}][2^{|E_s|}][|X||E| + |X|^2])$.

Proof. There are $(2(K+2))^{|X|}$ reduced information states. For each information state encountered, a maximum of $2^{|E_s|}$ control decisions are considered. For each control decision, we find the next Z state z, check if it satisfies the extended specification, and

for each $e \in C(z)$, find the next Y state. Finding z consists of computing the reduced unobservable reach and can be done in $O(|X||E|)$ time (see Appendix 1). Checking if it satisfies the extended specification can be done in time $O(|X|^2)$. Finally, there are at most $|E_o \cup E_s|$ successors to z, and each takes $O(|X|)$ time to compute since there are at most $2|X|$ augmented states in reduced information state $I(z)$, and we must consider the occurrence of event e from each of these to compute $h_{ZY}(z, e)$. The total running time is therefore $O([(2(K+2))^{|X|}][2^{|E_s|}][|X||E| + |X|^2 + |E_o \cup E_s||X|])$. But $|E_o \cup E_s| \leq |E|$, so this reduces to $O([(2(K+2))^{|X|}][2^{|E_s|}][|X||E| + |X|^2])$. $\square$

2.8.1 Experimental Results

We have implemented Algorithm 1 in C++ and run the program on randomly generated automata in order to demonstrate the scalability of the algorithm. We also created a program to generate random automata in accordance with the following specifications, in terms of input parameters n, ny, nm, nn, py, pm, pn, and pf:

- The generated automaton will have $|X| = n$ states, $|E_o| = ny$ events that are always observed, $|E_s| = nm$ events that may or may not be observed, $|E_{uo}| = nn$ non-faulty unobservable events, and $|E_f| = 1$ fault event e_f.
- The generated automaton will be accessible and will have no fault event self-loops.
- From any $x \in X$, the probability that $f(x, e)$ is defined is given by py if $e \in E_o$, pm if $e \in E_s$, pn if $e \in E_{uo}$, and pf if $e = e_f$. These probabilities are independent between one state and another, given the accessibility constraint.

We used the parameters $ny = 0$, $nm = 4$, $nn = 1$, $py = 0$, $pm = 0.35$, $pn = 0.05$, and $pf = 0.05$. These parameters were chosen by trial and error. We found that low probabilities tended to result in automata that were trivial or almost trivial (in some cases, it was safe to choose to monitor no events from the initial state). When

the probabilities were too high, the generated automata were undiagnosable. As the purpose of these results is to demonstrate the scalability of the algorithm, we do not present data with different parameters.

Table 2.1 shows MPO sizes and running times (rounded to the nearest second) for two versions of the program, one that makes use of the reduced information state and associated faster algorithms (e.g., for computed the reduced unobservable reach and for verifying if the extended specification is satisfied) and one that does not. We ran ten simulations each with $n = 75$ states and $n = 100$ states, respectively. We used $K = 4$ for the 75 state simulations and $K = 5$ for the 100 state simulations. One of the randomly generated automata with $n = 75$ states and two with $n = 100$ states were not K-diagnosable. The results of Table 2.1 show that the use of the reduced information state has little effect on the size of the resulting MPO, but that the algorithms for working with the reduced information state reduce running time by a factor of 3 or 4. We conjecture that the lack of structure of the randomly generated automata accounts for the small effect on MPO size of the reduced information state. The results of Table 2.1 also show that large MPOs can be generated in a small amount of time, which demonstrates that the algorithm of Sec. 2.8 is scalable in practice. All simulations were run on a 6GB Dell XPS 12 Laptop with a 1.6GHz intel core i7 processor (no parallelization was used in the simulations).

n	RIS?	MPO States/Time (s)									
		1	2	3	4	5	6	7	8	9	10
75	no	84325	N/A	86114	54502	54832	40404	29493	146108	169635	30819
		3	N/A	4	2	3	1	1	18	11	1
	yes	84258	N/A	86114	54502	54829	40325	29493	146003	169635	30674
		1	N/A	1	1	1	1	1	4	5	0
100	no	N/A	586886	871242	204583	149817	N/A	259381	199464	170652	2016398
		N/A	109	98	18	9	N/A	17	15	13	395
	yes	N/A	579455	871225	204458	149759	N/A	259380	199460	170618	2016395
		N/A	21	27	6	4	N/A	6	5	4	91

Table 2.1: Simulation results showing running times and MPO sizes with and without using the reduced information states, for randomly generated automata with 75 and 100 states.

2.9 Conclusion

This work considered the problem of dynamic fault diagnosis under the constraint of maintaining K-diagnosability. We established results about a structure called the MPO that contains all the solutions of the problem, developed from our notion of the information state. We next proved that the problem of finding safe controllers can be mapped to the state disambiguation problem, and showed an equivalence between safety and satisfying the extended specification. We proceeded to prove a monotonicity result on the extended specification that allowed us to reduce our information state and the size of our MPO. Putting all of our results together, we obtained a MPO with a size complexity of $O(2^{|X|}(K+2)^{|X|}2^{|E|})$, compared with $O(2^{|X|^2 \cdot K \cdot 2^{|E|}})$ for the previous approach of *Cassez and Tripakis* (2008). Finally, we presented an algorithm for computing the MPO, and demonstrated through simulations that the algorithm is scalable in practice. In future work, we will concentrate on finding a single optimal controller (according to some numerical cost criterion), using the MPO as a basis. We will also work on extending the results of this work to the problem of dynamic co-diagnosability.

Appendix 1: Computing the Extended Specification and the Reduced Unobservable Reach

Computing the Extended Specification

In this section, we show that computing the extended specification is equivalent to finding maximal weight paths on a particular graph. This idea was presented in *Cassez and Tripakis* (2008), in which the authors reduced the problem of finding the minimal K for which K-diagnosability can be achieved for a given automaton (i.e.,

the minimal detection delay). Computing the extended specification is a similar, but more general problem. In fact, in the notation to follow, the problem of computing this minimal K is equivalent to determining the single value $m^F(x_0, x_0)$. The method used here is very similar to that presented in *Yoo and Garcia* (2008) in that both approaches make use of strongly connected components to find minimal / maximal weight paths. The algorithm presented here differs from that of *Yoo and Garcia* (2008) in two ways: (i) we compute the extended specification rather than merely the minimal detection delay; and (ii) we use a customized algorithm for the problem of event-diagnosability whereas *Yoo and Garcia* (2008) considers the more general problem of language-diagnosability. Our algorithm therefore achieves a complexity of $O(|X|^2|E|)$, compared to $O(|X|^3|E|)$ as would be obtained by applying the method of *Yoo and Garcia* (2008). There are also similarities to the construction of the verifier *Yoo and Lafortune* (2002).

In what follows, let $L = \mathcal{L}(G)$ be the language of the automaton $G = (X, E, f, x_0)$ and let $L^{NF} = L \cap (E \setminus E_f)^*$, the language of G but excluding strings with fault occurrences. Also, for any state $x \in X$, let L/x denote the language $\mathcal{L}(G, x) = \mathcal{L}((X, E, f, x))$. That is, L/x denotes the language that is possible given the automaton G when starting in state x. Define L^{NF}/x analogously. Finally, for any string s and fault event e_f, let s/e_f denote the part of s beginning at the *first* occurence of event e_f in s, or ε if e_f does not occur in s. As before, we assume that there is only a single fault event, so that $E_f = \{e_f\}$.

In Sect. 2.6, we proved that safety is equivalent to satisfying the extended specification. By Thm. II.28, it suffices to find (for each $(x_1, x_2) \in X^2$) the minimum count n such that $((x_1, -1), (x_2, n)) \in T^e_{\text{spec}}$ to compute the extended specification. But by

Def. II.21, this is equivalent to finding:

$$\min \left\{ \begin{array}{c} n \in \{-1, 0, 1, \ldots\} : \exists s_1 \in L/x_1, s_2 \in L/x_2 \\ \text{s.t. } P(s_1) = P(s_2), N(g((x_1, -1), s_1)) = -1, \\ \text{and } N(g((x_2, n), s_2)) = K + 1 \end{array} \right\} \tag{2.24}$$

Instead of computing this minimum, we compute the following two maxima:

$$m(x_1, x_2) = \max_{\substack{s_1 \in L^{NF}/x_1, s_2 \in L/x_2 \\ \text{s.t. } P(s_1)=P(s_2)}} |s_2| \tag{2.25}$$

$$m^F(x_1, x_2) = \max_{\substack{s_1 \in L^{NF}/x_1, s_2 \in L/x_2 \\ \text{s.t. } P(s_1)=P(s_2)}} |s_2/e_f| \tag{2.26}$$

Theorem II.36.

$$\begin{aligned} &\min \left\{ \begin{array}{c} n \in \{-1, 0, 1, \ldots\} : \\ ((x_1, -1), (x_2, n)) \in T^e_{spec} \end{array} \right\} \\ &= \left\{ \begin{array}{ll} -1 & \textit{if } m^F(x_1, x_2) \geq K + 2 \\ \max\{0, K + 1 - m(x_1, x_2)\} & \textit{else} \end{array} \right. \end{aligned} \tag{2.27}$$

Proof. Suppose that $m^F(x_1, x_2) = c$. Then there exists $s_1 \in L^{NF}/x_1$ and $s_2 \in L/x_2$ such that $P(s_1) = P(s_2)$ and $|s_2/e_f| = c$. Since $s_1 \in L^{NF}/x_1$, it follows that $N(g((x_1, -1), s_1)) = -1$. Also, since $|s_2/e_f|$ is the number of events in s_2 starting at the first occurrence of e_f, it follows that $N(g((x_2, -1), s_2)) = c - 1$. Thus, the minimum in equation (2.24) is $n = -1$ if $c - 1 \geq K + 1$. Now suppose instead that $m^F(x_1, x_2) < K + 2$ and that $m(x_1, x_2) = c$. Then there exists $s_1 \in L^{NF}/x_1$ and $s_2 \in L/x_2$ such that $P(s_1) = P(s_2)$ and $|s_2| = c$. As before, this implies that $N(g((x_1, -1), s_1)) = -1$. It also implies that $N(g((x_2, n), s_2)) = c + n$ if $n \geq 0$ and

hence $N(g((x_2, n), s_2)) \geq K + 1$ if $n \geq 0$ and $n \geq K + 1 - m(x_1, x_2)$. $\square$

An advantage to using this method for computing the extended specification is that the values in equations (2.25) and (2.26) are not dependent on the particular value of K. This means that we do not need to recompute the extended specification for different values of K. The procedure for computing the values of $m(\cdot, \cdot)$ and $m^F(\cdot, \cdot)$ is described below. The two sets of values are computed by finding maximal weight (not necessarily simple) paths in a graph. The graph will be described below but it is effectively the complete (in the sense that we do not trim the non-accessible parts) one-sided verifier of G. The intuition for the procedure is that it generates all pairs of strings that have the same projection (with the first string being non-faulty), from each pair of states of the automaton G, and assigns a value to such a pair of strings equal to the length of the second string (in the case of the $m(\cdot, \cdot)$ values), or equal to the number of events starting from the first fault occurrence of the second string (in the case of the $m^F(\cdot, \cdot)$ values).

Step 1:

Create the graph $G_{es} = (V_{es}, A_{es})$, where $V_{es} = X \times \{N\} \times X \times \{N, Y\}$, and $A_{es} \subseteq V_{es} \times V_{es}$ is the set of (directed) edges between them, with labels in the set $E \times E$. Here, N and Y are fault labels, where N represents no fault (i.e., a count of -1) and Y represents the occurence of a fault at some point in the past (i.e., any count not equal to -1). We use $FL = \{N, Y\}$ to denote the set of fault labels. Define the function $FL_{es} : V_{es} \to FL$ to be the fault label associated with the second state of a vertex. That is:

$$FL_{es}(v) = \begin{cases} N, & \text{if } v \in X \times \{N\} \times X \times \{N\} \\ Y, & \text{if } v \in X \times \{N\} \times X \times \{Y\} \end{cases} \tag{2.28}$$

Also, let $EL_{es} : A_{es} \to E \times E$ be the function that assigns labels to edges. The set of edges A_{es} is defined by three cases:

- *Observed Events*: For any vertex $v_1 = (x_1, N, x_2, fl)$ and any event $e \in E_o \cup E_s$, if $f(x_1, e)$ and $f(x_2, e)$ are both defined, then there exists an edge $(v_1, v_2) \in A_{es}$ with label (e, e), where $v_2 = (f(x_1, e), N, f(x_2, e), fl)$.

- *Unobservable Events*: For any vertex $v_1 = (x_1, N, x_2, fl)$ and any event $e \in E_{uo}$, if $f(x_1, e)$ is defined, then there exists an edge $(v_1, v_2) \in A_{es}$ with label (e, ε), where $v_2 = (f(x_1, e), N, x_2, fl)$. Similarly, if $f(x_2, e)$ is defined, then there exists an edge $(v_1, v_2) \in A_{es}$ with label (ε, e), where $v_2 = (x_1, N, f(x_2, e), fl)$.

- *Faulty Events*: For any vertex $v_1 = (x_1, N, x_2, fl)$, if $f(x_2, e_f)$ is defined, then there exists an edge $(v_1, v_2) \in A_{es}$ with label (ε, e_f), where $v_2 = (x_1, N, f(x_2, e_f), Y)$.

Remark that, for any edge $(v_1, v_2) \in A_{es}$, its corresponding label (e_1, e_2) will satisfy $P(e_1) = P(e_2)$ in any of the above three cases. Next, we assign weights to each edge of G_{es} through the function $W_{es} : A_{es} \to \{0, 1\}$ as follows. For any $a = (v_1, v_2) \in A_{es}$ with label $EL(a) = (e_1, e_2)$,

$$W_{es}(a) = \begin{cases} 1, & \text{if } [FL_{es}(v_1) = N \text{ and } e_2 = e_f] \\ & \text{or } [FL_{es}(v_1) = Y \text{ and } e_2 \neq \varepsilon] \\ 0, & \text{else} \end{cases} \tag{2.29}$$

Thus $W_{es}(a) = 1$ precisely when it corresponds to an event occurrence that would increment the count on the x_2 state in the augmented automaton. For any $(x_1, x_2) \in X^2$, the value of $m(x_1, x_2)$ is equal to the maximal weight of a path starting at (x_1, N, x_2, Y). Similarly, the value of $m^F(x_1, x_2)$ is equal to the maximal weight of a path starting at (x_1, N, x_2, N). Let the weight of the maximal weight path starting from $v \in V_{es}$ be denoted by $d_{es}(v)$.

Step 2:

Create the component graph $G_{es}^{SCC} = (V_{es}^{SCC}, A_{es}^{SCC})$ of G_{es}, which is the graph over

strongly connected components (SCCs) of G_{es}. If we denote by $SCC(v)$ the SCC containing vertex v, we can write $V_{es}^{SCC} = \{SCC(v) : v \in V_{es}^{e}\}$ and $A_{es}^{SCC} = \{(S_1, S_2) \in V_{es}^{SCC} \times V_{es}^{SCC} : S_1 \neq S_2 \wedge \exists v_1 \in S_1, v_2 \in S_2 \text{ s.t. } (v_1, v_2) \in A_{es}\}$. Also, assign weights to each edge of G_{es}^{SCC} through the function $W_{es}^{SCC} : A_{es}^{SCC} \to \{0, 1\}$ as follows. For any $a = (S_1, S_2) \in A_{es}^{SCC}$,

$$W_{es}^{SCC}(a) = \max_{v_1 \in S_1, v_2 \in S_2 : (v_1, v_2) \in A_{es}} W_{es}((v_1, v_2)) \ . \tag{2.30}$$

Since all edge weights of G_{es} are non-negative, the maximal path weights starting from any two vertices in the same strongly connected components must be equal. That is, $SCC(v_1) = SCC(v_2) \Rightarrow d_{es}(v_1) = d_{es}(v_2)$. This is clearly true if all edges in a SCC have weight zero. If there is a non-zero edge, it can be traversed infinitely often on a maximal weight path starting from any vertex in the SCC, in which case the maximal path weights will be infinite for any vertex in the SCC. Thus, we can define $d_{es}^{SCC}(S)$ for any $S \in V_{es}^{SCC}$ and give it the value of $d_{es}(v)$ for any $v \in S$.

Step 3:

To compute the maximal weight paths, we consider the following three cases:

- For any $S \in V_{es}^{SCC}$ such that there exist two vertices $v_1, v_2 \in S$ and an edge $a = (v_1, v_2)$ with weight 1, assign $d_{es}^{SCC}(S) = \infty$, since this edge can be traversed an infinite number of times in the maximal weight path starting from any $v \in S$.

- For any $S_1 \in V_{es}^{SCC}$ that has a path to some $S_2 \in V_{es}^{SCC}$ in G_{es}^{SCC} with $d_{es}^{SCC}(S_2) = \infty$, assign $d_{es}^{SCC}(S_1) = \infty$, since there exists a path from any $v_1 \in S_1$ to any $v_2 \in S_2$ in this case (and hence the weight of the maximal weight path from any such v_1 is at least as great as that of the maximal weight path from any such v_2).

- For the remainder of the vertices of G_{es}^{SCC}, we can do a topological sort, assign

$d_{es}^{SCC}(S) = 0$, for any $S \in V_{es}^{SCC}$ that is a sink of G_{es}^{SCC} (i.e., with no outgoing edges), and work backwards from these to compute the remaining values of $d_{es}^{SCC}(S)$. Considering the vertices in this order ensures that we find the longest path from any vertex before moving on to any of its predecessors in the topological sort.

Proposition II.37 (Running Time of Extended Specification Computation)**.** *The running time of the procedure described in this section is $O(|X|^2|E|)$ if G is deterministic.*

Proof. The graph created in step 1 has $|V_{es}| = 2|X|^2$ vertices. For a deterministic automaton, there is at most one transition defined for a given initial state and event. Thus, for each of the $2|X|^2$ vertices of G_{es}, there is at most one outgoing edge for each event $e \in E_o \cup E_s$ (labeled (e, e)), at most two outgoing edges for each event $e \in E_{uo}$ (labeled either (e, ε) or (ε, e)), and at most one outgoing edge for each event $e \in E_f$ (labeled (ε, e)). Hence, the graph created in step 1 has $|A_{es}| \leq [2|X|^2][2|E|] = 4|X|^2|E|$ edges. Creating the component graph in step 2 can be done in time $O(|V_{es}| + |A_{es}|)$ (see e.g., *Cormen et al.* (2009)). Furthermore, $|V_{es}^{SCC}| \leq |V_{es}| = 2|X|^2$ and $|A_{es}^{SCC}| \leq |V_{es}^{SCC}| - 1 < 2|X|^2$. Computing W_{es}^{SCC} and performing the first part of step 3 can be done together in time $O(|A_{es}|)$, by considering each edge $(v_1, v_2) \in A_{es}$ such that $W_{es}(v_1, v_2) = 1$. If $SCC(v_1) = SCC(v_2) = S$, then we set $d_{es}^{SCC}(S) = \infty$. If $SCC(v_1) \neq SCC(v_2)$, then we set $W_{es}^{SCC}(SCC(v_1), SCC(v_2)) = 1$. The second part of step 3 can be done through a single depth-first search on G_{es}^{SCC}, and hence takes $O(|X|^2)$ time. For the last part of step 3, finding a topological sort can be done at the same time as computing the component graph. The remainder of the algorithm takes linear time in the size of G_{es}^{SCC} by considering this graph's vertices in topologically sorted order, starting from sink nodes. Thus, the total running time is in $O(|X|^2|E|)$. □

Computing the Reduced Unobservable Reach

In this section, we show how to efficiently compute the reduced unobservable reach (i.e., how to compute $rur = RUR(y, \gamma)$). The naive method of computing the unobservable reach and reducing the resulting information state has a running time of $O(K|X||E|) + O(|X|)$, which is worse than the $O(|X||E|)$ running time of the algorithm presented below. The procedure for this computation bears some similarity to the one used to compute the extended specification in that we also make use of strongly connected components and the topological sort. In what follows, we assume that the computation of the unobservable reach is a step in determining whether or not a particular control decision is safe, so that we stop immediately if we find an augmented state with a count of more than $K + 1$ in the unobservable reach.

Step 1: Graph construction

Create the graph $G_\gamma = (X, A_\gamma)$, where $A_\gamma \subseteq X \times X$ is the set of (directed) edges. Let $A_\gamma = A_\gamma^F \cup A_\gamma^{NF}$, where these are defined by:

$$\begin{aligned} A_\gamma^F &= \{(x_1, x_2) \in X^2 : \exists e \in E_f \text{ s.t. } f(x_1, e) = x_2\} \\ A_\gamma^{NF} &= \left\{ \begin{array}{c} (x_1, x_2) \in X^2 : \exists e \in E \setminus (\gamma \cup E_f) \\ \text{s.t. } f(x_1, e) = x_2 \end{array} \right\} \end{aligned}$$

Thus, the set of edges A_γ corresponds simply to all unobservable transitions of the automaton G, given the set of monitored events γ, and are split into A_γ^F (for fault transitions) and A_γ^{NF} (for non-faulty transitions). Note that some (x_1, x_2) may be in both sets.

Step 2: Finding the -1 count augmented states

Initialize $rur \leftarrow y$. For all $u \in y$ such that $N(u) = -1$, determine all $x \in X$ such

that there exists a path from $S(u)$ to x in G_γ, considering only edges in A_γ^{NF}, and set $rur \leftarrow rur \cup \{(x, -1)\}$. This can be done through a single depth-first search on the graph G_γ and gives the set of all $v \in RUR(y, \gamma)$ such that $N(v) = -1$.

Step 3: Finding the 0 count augmented states

For each $u \in rur$ such that $N(u) = -1$ and for each $x \in X$ such that $(S(u), x) \in A_\gamma^F$, set $rur \leftarrow rur \cup \{(x, 0)\}$, unless there exists some $v \in rur$ such that $v >^+ (x, 0)$.

Step 4: Finding the maximal count augmented states

Create the component graph $G_\gamma^{SCC} = (V_\gamma^{SCC}, A_\gamma^{SCC})$ of G_γ defined over strongly connected components. For each $x \in X$, let $SCC_\gamma(x)$ denote the SCC that contains x. We first check if there are any augmented states that can be reached with arbitrarily large count. This will occur if there exists some $x \in X$ such that $(x, n) \in rur$ for some $n \geq 0$ and a path from $SCC_\gamma(x)$ to some SCC $S \in V_\gamma^{SCC}$ with $|S| > 1$. In this case, we stop here and determine that γ was not a safe control decision. If we do not halt at this point, then we know that all non-singleton SCCs are unreachable by any $x \in X$ such that $(x, n) \in rur$ for some $n \geq 0$. It follows that any non-singleton SCCs in G_γ^{SCC} will not have any effect on the maximal counts. Therefore, we remove all these states from G_γ^{SCC}, obtaining a subgraph G'_γ of G_γ. Finally, we compute the maximal counts by considering the vertices in topologically sorted order, starting from source nodes. This guarantees that we will find all paths to a vertex x (and hence the maximal value of n such that $(x, n) \in RUR(y, \gamma)$) before moving on to any of its successors.

Proposition II.38 (Running Time of Reduced Unobservable Reach Computation)**.** *The running time of the procedure described in this section is $O(|X||E|)$ if G is deterministic.*

Proof. The graph created in step 1 has $|X|$ vertices and at most $|E \backslash \gamma| \leq |E|$ outgoing

edges per vertex and can therefore be constructed in time $O(|X||E|)$. The depth-first search in step 2 takes linear time in the size of the graph G_γ. Step 3 requires considering all fault transitions in the automaton. Since G is deterministic, and there is only one fault event, there are at most $|X|$ such transitions and hence this step is done in $O(|X|)$ time. In step 4, computing the component graph can be done in linear time in the size of the graph G_γ (see e.g., *Cormen et al.* (2009)). We can then find all the SCCs of G_γ^{SCC} that are not singletons and mark them in time $O(|X|)$. Checking if there exists a path to one of these non-singleton SCCs from some $x \in X$ such that $(x, n) \in rur$ for some $n \geq 0$ can be done with a single depth-first search on G_γ^{SCC}, which takes linear time. Removing the non-singleton SCCs from G_γ^{SCC} and topologically sorting the remaining graph also takes linear time in the size of G_γ^{SCC}. Finally, computing the maximal counts by considering the vertices in topologically sorted order takes linear time in the size of G'_γ. Since all of the operations are linear time and no graph has size larger than $O(|X||E|)$, the overall running time is $O(|X||E|)$. □

Appendix 2: Proofs not contained in main body

Proof of Lemma II.11:

Proof. The proof is established by induction on the length of $P_C(s)$. Let $|P_C(s)| = n$. Furthermore, for any string t, let $t[k]$ denote the k^{th} event in t, and let $t(k)$ denote the substring $t[1] \cdots t[k]$, with $t(0) = \varepsilon$. As further shorthand, let $s_k = P_C(s)(k)$ for $k = 0, \ldots, n$ and $e_k = P_C(s)[k+1]$ for $k = 0, \ldots, n-1$, so that $s_0 = \varepsilon$, $s_1 = e_0$, etc... Define y_0 as usual. For $k = 0, \ldots, n$, let $z_k = h_{YZ}(y_k, C(y_k))$ and for $k = 0, \ldots, n-1$, define $y_{k+1} = h_{ZY}(z_k, e_k)$. Finally, for $k = 0, \ldots, n$, define C_k by $C_k = C(y_k)$. First, notice that since unobserved events do not change the information state, we have $IS_C^Z(s_k) = z_k$ and, in particular, $z = IS_C^Z(s) = IS_C^Z(P_C(s)) = IS_C^Z(s_n) = z_n$. The

inductive hypothesis is:

$$I(z_k) = \{v \in X^+ : \exists s'_k \text{ s.t. } P_C(s'_k) = s_k \wedge v = g(s'_k)\}, \tag{2.31}$$

where we have dropped the $P_C(\cdot)$ around s_k since s_k is already a projection. For the base case z_0, we have that:

$$\begin{aligned} I(z_0) &= UR(y_0, C_0) \\ &= \left\{ \begin{array}{l} v \in X^+ : (\exists u \in y_0)(\exists t \in (E \setminus C_0)^*) \\ \quad \text{s.t. } v = g(u,t) \end{array} \right\} \\ &= \{v \in X^+ : \exists t \in (E \setminus C_0)^* \text{ s.t. } v = g(x_0^+, t)\} \\ &= \{v \in X^+ : \exists t \text{ s.t. } P_C(t) = \varepsilon = s_0 \wedge v \in g(t)\} \end{aligned}$$

Thus the base case is established, by taking $s'_0 = t$. Now suppose that the inductive

hypothesis is true at k. Then:

$$\begin{aligned}
y_{k+1} &= h_{ZY}(z_k, e_k) \\
&= \{v \in X^+ : \exists u \in I(z_k) \text{ s.t. } v = g(u, e_k)\} \\
&= \left\{ v \in X^+ : \begin{array}{c} \exists s'_k \text{ s.t. } P_C(s'_k) = s_k \\ \text{and } v = g(s'_k e_k) \end{array} \right\} \\
I(z_{k+1}) &= UR(y_{k+1}, C_{k+1}) \\
&= \left\{ v \in X^+ : \begin{array}{c} (\exists u \in y_{k+1})(\exists t \in (E \setminus C_{k+1})^*) \\ \text{s.t. } v = g(u, t) \end{array} \right\} \\
&= \left\{ v \in X^+ : \begin{array}{c} (\exists s'_k)(\exists t \in (E \setminus C_{k+1})^*) \\ \text{s.t. } P_C(s'_k) = s_k \\ v = g(s'_k e_k t) \end{array} \right\} \\
&= \left\{ v \in X^+ : \begin{array}{c} \exists s'_{k+1} \text{ s.t. } P_C(s'_{k+1}) = s_{k+1} \\ \text{and } v = g(s'_{k+1}) \end{array} \right\},
\end{aligned}$$

where the last equality follows by taking $s'_{k+1} = s'_k e_k t$ and noting that, since s'_k can be any string satisfying $P_C(s'_k) = s_k$ and t can be any string satisfying $t \in (E \setminus C_{k+1})^*$ (which is equal to the set $\{t : P_C(z_k, t) = \varepsilon\}$), the concatenation $s'_{k+1} = s'_k e_k t$ can be any string satisfying $P_C(s'_{k+1}) = s_k e_k = s_{k+1}$. Thus the induction step is proven and the lemma follows from this. □

CHAPTER III

Vehicle Control : The case of perfect measurement

3.1 Abstract

We consider the problem of controlling a set of vehicles at an intersection, in the presence of uncontrolled vehicles and a bounded disturbance. We begin by discretizing the system in space and time to construct a suitable discrete event system (DES) abstraction, and formally define the problem to be solved as that of constructing a supervisor over the discrete state space that is safe (i.e., collision-free), non-deadlocking (i.e., the vehicles all cross the intersection eventually), and maximally permissive with respect to the chosen discretization. We show how to model the uncontrolled vehicles and the disturbance through uncontrollable events of the DES abstraction. We define two types of relations between systems and their abstraction: state reduction and exact state reduction. We prove that, when the abstraction is a state reduction of a continuous system, then we can obtain a safe, non-deadlocking, and maximally permissive memoryless supervisor. This is obtained by translating safety and non-deadlocking specifications to the abstract domain, synthesizing the supervisor in this domain, and finally translating the supervisor back to the concrete domain. We show that, when the abstraction is an exact state reduction, the resulting supervisor will be maximally permissive among the class of all supervisors, not merely memoryless ones. Finally, we provide a customized algorithm and demonstrate its scalability through

simulation.

3.2 Introduction

We consider the problem of controlling a set of n vehicles in the vicinity of an intersection. We assume that vehicles move along a set of m intersecting two-way roads, $m \leq n$, and that the path that each vehicle will follow is known a priori (for example, by means of reading the turn signal of the vehicle), and we want to supervise the vehicles' behaviour to avoid a side impact of any two vehicles on intersecting paths, and a rear-end collision of any two vehicles on a common or on merging paths. See Fig. 3.1 for an example.

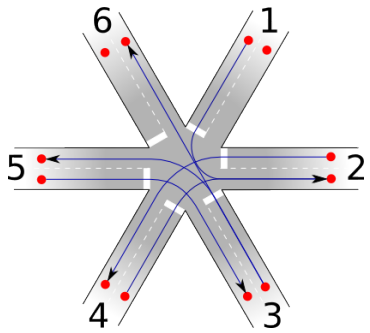


Figure 3.1: An example of the vehicle control problem.

We assume that a certain subset of the vehicles are uncontrolled, and that there is a disturbance on the vehicle dynamics with a known bound. The problem to be solved consists of designing a supervisor that restricts the actions of the controlled vehicles such that the system is safe (i.e., collision-free), non-deadlocking (i.e., the vehicles must eventually cross the intersection), and maximally permissive.

Three common approaches to this problem include: the computation of maximally controlled invariant sets; mapping the problem to that of scheduling; and abstraction/symbolic models. Among approaches falling in the first category, we mention, e.g., (*Hafner and Del Vecchio*, 2011; *Verma and Del Vecchio*, 2011; *Hafner et al.*,

2013). By explicitly computing the *capture set*, or set of states from which it is not possible to guarantee avoidance of the unsafe states, these approaches naturally satisfy safety, non-deadlockingness and maximal permissiveness, and can deal with sources of uncontrollability and also with measurement uncertainty. However, such approaches typically require conditions on the geometry of the unsafe set and on the structure of the dynamics, or else scale poorly to systems with more than a few dimensions. See also (*Tomlin et al.*, 2003) for an example involving a flight management system. Scheduling approaches work by allocating time intervals during which the vehicles can be inside the intersection. The scheduling problem is generally NP-hard but takes polynomial time in the special case where all jobs require the same processing time. Reducing the vehicle control problem to the polynomial-time scheduling case amounts to either an assumption of certain symmetries in the vehicle control problem set-up, or a problem relaxation where such symmetries are not satisfied. Approaches in this category include (*Colombo and Del Vecchio*, 2012), its extension to the case of dynamics with disturbances, (*Bruni et al.*, 2013), and its extension to the case of uncontrolled vehicles *Ahn et al.* (2014). Because of the assumption of mutual exclusiveness of the use of the intersection, these approaches do not deal with vehicles on non-intersecting paths (in which case multiple vehicles could simultaneously be in the intersection). Another approach is to pre-compute fail-safe maneuvers as in (*Kowshik et al.*, 2011), or evasion plans as in (*Au et al.*, 2012). These last approaches deal with some types of environmental uncertainty, but do not guarantee maximal permissiveness.

Our approach falls in the category of abstraction/symbolic models. Abstraction based methods work by mapping the continuous system model and specifications to a finite model and solving for a supervisor on the finite model, in such a way that the obtained supervisor can be used on the original (continuous) system, while preserving safety and non-deadlocking properties. Work in this domain includes (*Alur*

et al., 2000; *Daws and Tripakis*, 1998) in the context of verification / model checking, as well as (*Colombo and Del Vecchio*, 2011a,b; *Colombo and Girard*, 2013), which make use of differential flatness of dynamical systems to construct abstractions with provable errors bounds. Our work is most closely related to that of (*Girard et al.*, 2010; *Pola and Tabuada*, 2009; *Zamani et al.*, 2012; *Camara et al.*, 2011), which construct symbolic models that satisfy simulation or alternating simulation relations with the original system. In particular, this work also makes use of alternating simulation relations, and variations thereof.

In this problem, the number of vehicles will typically be at least five (we provide simulation results for six vehicles) and the bad set has a non-convex shape, which makes exact computation of the capture set intractable. On the other hand, the scheduling methods of (*Colombo and Del Vecchio*, 2012), (*Bruni et al.*, 2013), and (*Ahn et al.*, 2014) do not explicitly pre-compute sets of states from which there exist solutions to the corresponding scheduling problems, but instead perform verification on-line. Because the exact verification problem is NP-hard, only the polynomial-time problem relaxations are feasible in practice. While also suffering from problems related to state space explosion, abstraction based methods nevertheless offer more scalability than capture set computation and more flexibility than reductions to scheduling problems.

We proceed to solve the problem by discretizing the system in space and time, thus obtaining a finite solution space. Using this discretization as a basis, we construct a discrete-event system (DES) abstraction and model the two sources of uncontrollability (the uncontrolled vehicles and the disturbance) through uncontrollable events. By translating the safety and non-deadlocking specifications from the continuous to the discrete-event domain, we formulate the problem to be solved in the context of supervisory control theory of DES (see (*Ramadge and Wonham*, 1987), (*Wonham*, 2013), (*Cassandras and Lafortune*, 2008)). Specifically, we obtain a maximally permissive

safe and non-deadlocking supervisor for the DES by solving the Basic Supervisor Control Problem in the Non-Blocking case (BSCP-NB). The resulting supervisor is then translated back to the original (continuous) problem domain, preserving safety, non-deadlockingness, and maximal permissiveness with respect to the discretization.

To prove that safety and non-deadlockingness are preserved when translating the obtained supervisor from the abstract back to the continuous problem domain and to characterize the sense in which the resulting solution is maximally permissive, we define two types of relations between systems and their abstractions: the state reduction and the exact state reduction. We prove that, when the abstraction is a state reduction of the original system, the obtained supervisor for the continuous domain problem will be safe, non-deadlocking, and maximally permissive among the class of memoryless supervisors. When the abstraction is an exact state reduction of the original system, the obtained supervisor will be maximally permissive among the class of all supervisors, not merely memoryless ones. In the context of the vehicles control problem, we show that our DES abstraction is a state reduction of the continuous system model. Additionally, we show that, if the bounds on the disturbance are an integral multiple of one of the discretization parameters, then our DES abstraction becomes an exact state reduction of the continuous system model.

Finally, we present a customized algorithm to construct the supervisor for the vehicle control problem. By making use of the problem's structure, we are able to obtain an algorithm that is faster than the standard DES supervisory control algorithms. We show through simulation that the algorithm is scalable in practice, with running times of under one minute for systems with hundreds of millions of accessible transitions in the DES abstraction.

Our contributions are as follows. First, the translation of the system model and specifications to the domain of DES allows us to leverage methods from supervisory control theory, methods which are well-suited to finding maximally permissive su-

pervisors in the presence of uncontrolled elements of the environment. Second, the notions of state reduction and exact state reduction are general notions that conserve maximal permissiveness, rather than merely safety and non-deadlockness, when going from an abstraction back to the original system. Finally, the customized algorithm presented in this work makes use of system properties that could also generalize to other problems of interest. Preliminary versions of some of the results presented here have appeared in (*Dallal et al.*, 2013a), (*Dallal et al.*, 2013b).

The organization of this work is as follows. In Sec. 3.3, we present the system model, its time/space discretization, and the problem to be solved. In Sec. 3.4, we describe the set of collision points to be avoided. In Sec. 3.5, we define the DES abstraction of the system defined in Sec. 3.3. In Sec. 3.6, we present the state reduction, exact state reduction, and associated theorems. In Sec. 3.7, we provide an overview of supervisory control theory, prove that the abstraction defined in Sec. 3.5 is a state reduction of the system defined in Sec. 3.3, and additionally prove under what conditions the abstraction is an exact state reduction. In Sec. 3.8, we present our customized algorithm for solving the vehicle control problem. In Sec. 3.9, we present simulation results for an implementation of our algorithm. Finally, we conclude in Sec. 3.10. We also include derivations of the equations used in our algorithms, which are contained in the appendix.

3.3 Model and Problem Definition

Consider a set of n vehicles $\mathcal{N} = \{1, \ldots, n\}$ modeled as kinematic entities (integrators) and described by

$$\dot{x} = v + d \tag{3.1}$$

where $x \in X \subset \mathbb{R}^n$ is the state, $v \in V \subset \mathbb{R}^n$ is the control input, and $d \in D \subset \mathbb{R}^n$ is a disturbance input representing unmodeled dynamics (for instance, the dynamic

response of the vehicle to the engine torque). Assume that X is compact (i.e., the vehicles are controlled in some neighbourhood of the intersection) and that $D = [d_{min}, d_{max}]^n$, with $d_{min} \leq 0 \leq d_{max}$. We take the set V to be the (discrete) set of vectors with elements in the finite set $\{a\mu, (a+1)\mu, \ldots, b\mu\}$, with $a, b \in \mathbb{N}$ and $\mu \in \mathbb{R}_+$. The values $a\mu$ and $b\mu$ are denoted by v_{min} and v_{max}, respectively. To allow for the possibility that a subset of the vehicles cannot be controlled, let v be partitioned into two subvectors, $v_c \in V_c$ for the controlled vehicles, and $v_{uc} \in V_{uc}$ for the uncontrolled vehicles, so that $v = (v_c, v_{uc})$ and $V = V_c \times V_{uc}$. Assume also that $v_{min} + d_{min} \geq \mu$, so that μ constitutes a lower bound on the velocity of the vehicles. Finally, assume that the input v is kept constant over time intervals $[k\tau, (k+1)\tau]$, $k \in \mathbb{N}$ and discretize the above system in time with step τ, obtaining

$$x_{k+1} = x_k + u_k + \delta_k \tag{3.2}$$

with $x_k = x(k\tau)$, $u_k = v(k\tau)\tau$, $\delta_k = \int_{k\tau}^{(k+1)\tau} d(t)dt$. Calling $U = V\tau$ and $\Delta = D\tau$, we have that $u \in U$ and $\delta \in \Delta$. In the remainder of this work, we will also use the notation $\delta_{\min} := d_{\min}\tau$ and $\delta_{\max} = d_{\max}\tau$. As with the set V, we use the notation $u = (u_c, u_{uc})$ to denote the controls of the controlled and uncontrolled vehicles and write $U = U_c \times U_{uc}$. Next, we discretize the system in space by defining a set of discrete states $\tilde{Q}$ and a mapping $\ell : X \to \tilde{Q}$ from continuous to discrete states as follows:

$$\ell_i(x_i) := \begin{cases} c\tau\mu, \text{ for } c \in \mathbb{Z} \text{ s.t.} & \\ c\tau\mu - \tau\mu/2 < x_i \leq c\tau\mu + \tau\mu/2, & \text{if } x_i \leq \alpha_k \\ q_{i,m}, & \text{if } x_i > \alpha_k \end{cases} \tag{3.3}$$

where k is the index of the road on which vehicle i exits the intersection (i.e., after any turn) and α_k marks the end of the intersection on road k (the shape of the intersection will be described in more detail in Sec. 3.4). Note that, if the vehicles are to be controlled beyond the end of the intersection, then a value greater than α_k could

be used in Eq. (3.3). This could potentially result in more than one marked state in the definition of G (see Sec. 3.5) and would not invalidate any results presented in this work. Define $\ell(x)$ as the vector $(\ell_1(x_1), \ldots, \ell_n(x_n))$ and define the notation $\ell^{-1}(q) = \{x \in X : \ell(x) = q\}$. In words, the space X is covered by a regular lattice with spacing $\tau\mu$. Vehicles before the end of the intersection are mapped to a point of this lattice whereas vehicles after the end of the intersection are mapped to "special" states $q_{i,m}$. The state $q_m = (q_{1,m}, \ldots, q_{n,m})$ is the (unique) discrete state where all vehicles have crossed the intersection. Assume that, for all $q \in \tilde{Q}$, there exists some $x \in X$ such that $\ell(x) = q$. Finally, assume that there is some set B of bad states (representing collision points) and that we would like to define a supervisor so that $x(t) \notin B \, \forall \, t \geq 0$. We will describe the bad set in the following section. Specifically, we wish to solve the following problem:

Problem III.1. Let X/ℓ denote the quotient set of X with respect to the equivalence relation $E \subseteq X \times X$ defined by $(x_1, x_2) \in E \Leftrightarrow \ell(x_1) = \ell(x_2)$. Given $\tilde{Q}$, define a supervisor $\sigma : X/\ell \to 2^{V_c}$ that assigns to each $x(k\tau) \in X$ a set of inputs $v_c \in V_c$ allowed for the interval $[k\tau, (k+1)\tau]$ and constant over this time interval, with the following properties:

- if $v_c(t) \in \sigma(x(\lfloor t/\tau \rfloor \tau))$ for $t \in [k\tau, (k+1)\tau]$, then $x(t) \notin B$ in the same time interval (safety)
- if $\sigma(x(k\tau)) \neq \emptyset$, $v_c(t) \in \sigma(x(\lfloor t/\tau \rfloor \tau))$ for $t \in [k\tau, (k+1)\tau]$, and $\ell(x((k+1)\tau)) \neq q_m$, then $\sigma(x((k+1)\tau)) \neq \emptyset$ (non-deadlockingness)
- if $\tilde{\sigma} \neq \sigma$ and $\tilde{\sigma}$ satisfies the two properties above, then $\tilde{\sigma}(x(k\tau)) \subseteq \sigma(x(k\tau))$ for all $k \geq 0$ (maximal permissiveness). $\square$

3.4 Bad Set Description

Let the set of roads in this system be denoted by $\mathcal{R} = \{1, \ldots, m\}$. Associated to each vehicle i is a pair of roads $(r_{i,1}, r_{i,2})$, indicating that the vehicle starts on road $r_{i,1}$ and turns onto road $r_{i,2}$ at the intersection. Each road r in this system is parametrized by the length α_r of the road that is inside the intersection. We assume that vehicles instantaneously switch from one road to another (i.e., when turning) at point 0. Thus, vehicle i is on road $r_{i,1}$ when $x_i < 0$, inside the intersection when $x_i \in [-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}]$, and on road $r_{i,2}$ when $x_i > 0$. We define any two pairs of roads $(r_{i,1}, r_{i,2})$ and $(r_{j,1}, r_{j,2})$ as *conflicting* pictorially as follows.

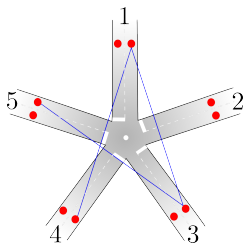


Figure 3.2: An example scenario involving three vehicles on five roads. Blue lines are drawn for each vehicle indicating starting road and ending road.

Let each road $r \in \mathcal{R}$ be represented by two points (one for each direction of traffic), arranged as a regular polygon with m sides. Figure 3.2 shows an example with five roads. Now suppose that vehicles travel on the right side of the road and represent a pair of roads (r_1, r_2) by a line segment from the right point of road r_1 (when looking toward the center of the intersection) to the right point of road r_2 (when looking away from the center of the intersection). We say that the two pairs of roads $(r_{i,1}, r_{i,2})$ and $(r_{j,1}, r_{j,2})$ are conflicting if their respective line segments intersect. We identify two types of constraints, depending on whether the intersection point is an endpoint of the line segments (case 1) or not (case 2). The former case (case 1) will occur if two vehicles are travelling on the same road and in the same direction, either before or after reaching the origin. Figure 3.2 contains two examples of this: there are two

vehicles entering the intersection on road 3, and there are two vehicles exiting the intersection on road 1. In this case, we place the safety constraint that they must maintain a distance of at least γ, as long as both vehicles are on this road. Thus, if $r_{i,1} = r_{j,1}$, then the forbidden set of points is given by $|x_i - x_j| < \gamma$, $x_i, x_j \leq 0$. Similarly, if $r_{i,2} = r_{j,2}$, then the forbidden set of points is given by $|x_i - x_j| < \gamma$, $x_i, x_j \geq 0$. The latter case (case 2) will occur when there is a danger of a crash at the intersection between vehicles while turning. Figure 3.2 shows an example of this in which one vehicle turns from road 3 to road 5 and another turns from road 4 to road 1. In this case, we place the safety constraint that they must not simultaneously be in the intersection. Thus, the forbidden set of points is given by $-\alpha_{r_{i,1}} < x_i < \alpha_{r_{i,2}}$ and $-\alpha_{r_{j,1}} < x_j < \alpha_{r_{j,2}}$. If the line segments for two vehicles do not intersect, then no constraints are placed on the joint position of the two vehicles (ex: two vehicles on different roads turning onto the roads to their immediate rights). We call the set of all forbidden points the bad set, and denote it by B. Note that we do not include collision points involving two uncontrolled vehicles in the bad set, since these cannot be prevented through any control action.

3.5 Discrete Abstraction

In this section, we define a discrete event system (DES) and proceed to construct a DES G that models the behavior of the continuous time system, using the lattice $\tilde{Q}$ as the set of discrete states.

Definition III.2 (Discrete Event System)**.** A (deterministic) discrete event system is tuple $G = (X, E, \psi, x_0, X_m)$ where X is a set of states, E is a set of events, $\psi : X \times E \to X$ is a partial transition function, $x_0 \in X$ is the initial state, and $X_m \subseteq X$ is a set of marked states representing the completion of some behavior of interest.

To construct a DES abstraction of the continuous-time system, we use a three-layered transition function ψ. The first layer consists of events in the set U_c, for the actions of the controlled vehicles. The second layer consists of events in the set U_{uc}, for the actions of the uncontrolled vehicles. It remains to model the disturbance d. We achieve this by discretizing the set Δ to obtain a set of "discretized disturbances" W. Specifically, let

$$W = \{k\tau\mu : k \in \mathbb{Z} \wedge \lfloor \delta_{min}/(\tau\mu) \rfloor \leq k \leq \lceil \delta_{max}/(\tau\mu) \rceil\}^n. \tag{3.4}$$

This set W makes up the third layer of G's transition structure. For any $q \in \tilde{Q}$, $u_c \in U_c$, $u_{uc} \in U_{uc}$, and $w \in W$, we define

$$\psi(q, u_c u_{uc} w) := q + u + w, \tag{3.5}$$

where $u = (u_c, u_{uc})$. In Sec. 3.7, we will show that $\psi(q, u_c u_{uc} w) = q'$ if and only if there exist $x \in X$, $\delta \in \Delta$, and $x' \in X$ such that $x' = x + u + \delta$, $q = \ell(x)$, and $q' = \ell(x')$ (see Prop. III.20). To define the discrete system state in between the occurrence of events in U_c and U_{uc} and in between the occurrence of events in U_{uc} and W (all of which occur simultaneously in the continuous-time system), we introduce two sets of "intermediate" states Q_{I1} and Q_{I2} (disjoint from each other and from $\tilde{Q}$ and with no physical meaning), and three intermediate transition functions: $\psi_1 : \tilde{Q} \times U_c \to Q_{I1}$, $\psi_2 : Q_{I1} \times U_{uc} \to Q_{I2}$, and $\psi_3 : Q_{I2} \times W \to \tilde{Q}$, defined only by $\psi(q, u_c, u_{uc}, w) = \psi_3(\psi_2(\psi_1(q, u_c), u_{uc}), w)$. See Fig. 3.3 for a depiction of the transition function ψ. We take the set of marked states to be the set $Q_m = \{q_m\}$. Finally, we define a set Q_0 of possible initial states, which we model by introducing a dummy initial state q_0 and having transitions from q_0 to each state $q \in Q_0$ with event label e_q. We denote this set of events by $E_Q := \{e_q : q \in Q_0\}$ and define $\psi(q_0, e_q) := q$. The final DES is defined as:

$$G := (Q, E_Q \cup U_c \cup U_{uc} \cup W, \psi, q_0, Q_m) \tag{3.6}$$

where $Q = \{q_0\} \cup \tilde{Q} \cup Q_{I1} \cup Q_{I2}$. The sets of events U_c is taken to be controllable, whereas the sets of events U_{uc} and W are taken to be uncontrollable. Note that, in the context of supervisory control problems of DES, a supervisor is obtained which does not choose a particular event from any given state, but rather chooses which events to enable (allow) and which ones to disable (prevent). An uncontrollable event is an event that cannot be disabled.

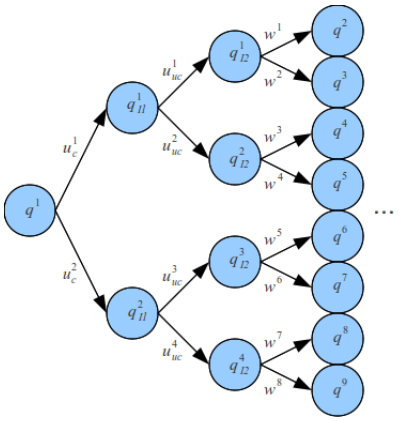


Figure 3.3: The transition function ψ.

Remark III.3. Although the initial state can not be chosen by the system, we take the set of events E_Q to also be controllable. In Sec. 3.7, we will use G as the basis for a supervisory control problem. If E_Q were defined as uncontrollable, we would obtain an empty solution to the supervisory control problem whenever there was *any* initial state from which there was no solution, even if there existed solutions from some of them. By defining the set E_Q as controllable, the computed supervisor will contain a transition from q_0 to q for every $q \in Q_0$ from which there exists a solution to the supervisory control problem. We will revisit this issue in Sec. 3.8.

3.6 State Reductions and Supervisory Control

In this section, we define two types of relations between systems: state reductions and exact state reductions, and prove theorems relating safety, non-deadlockness, and maximal permissiveness of supervisors for systems related through state reductions and exact state reductions. The state reduction and exact state reduction relations are based on the notions of alternating similarity relations, as defined in *Tabuada* (2009). The theorems proven in this section will be used later in this work to establish the correctness of our solution to Prob. III.1. We begin with some preliminary definitions.

3.6.1 Preliminaries

Definition III.4 (System)**.** A system S is defined as a tuple $S = (X, U, \rightarrow, Y, H)$, where X is the set of states, U is a set of control inputs, $\rightarrow \subseteq X \times U \times X$ is a transition relation, Y is an output set, and $H : X \rightarrow Y$ is the output function. □

For a system $S = (X, U, \rightarrow, Y, H)$, we will use the notation $\mathbf{Post}_u(x) := \{x' \in X : (x, u, x') \in \rightarrow\}$ and $U(x) := \{u \in U : \mathbf{Post}_u(x) \neq \emptyset\}$. In the remainder of this work, it will be assumed that all systems satisfy the property $H(x_1) = H(x_2) \Rightarrow U(x_1) = U(x_2)$, for all $x_1, x_2 \in X$. In words, this means that any two states with the same observation should not be distinguishable by their available set of inputs.

Definition III.5 (Run)**.** A run ρ of length n for a system $S = (X, U, \rightarrow, Y, H)$ is a sequence of past states and inputs $(x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n)$, such that $u^i \in U(x^i)$ and $x^{i+1} \in \mathbf{Post}_{u^i}(x^i)$ for $i = 0, \ldots, n-1$. The set of runs of length n is denoted by $R_n(S)$ and the set of runs is $R(S) = \bigcup_{i=0}^{\infty} R_n(S)$. Given run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n)$, we define the notation $tgt(\theta) := x^n$ and $\rho(k) := (x^0, u^0, \ldots, x^{k-1}, u^{k-1}, x^k)$, called a *prefix* of ρ. □

Definition III.6 (History)**.** A history θ of length n for a system $S = (X, U, \rightarrow, Y, H)$ is a sequence of past outputs and inputs $(y^0, u^0, \ldots, y^{n-1}, u^{n-1}, y^n)$, such that there

exists a run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n) \in R_n(S)$ that is consistent with θ, in the sense that $y^i = H(x^i)$ for $i = 0, \ldots, n$. The set of histories of length n is denoted by $\Theta_n(S)$ and the set of histories is $\Theta(S) = \bigcup_{i=0}^{\infty} \Theta_n$. We will also write $\theta(\rho)$ to mean the unique history produced by a run $\rho \in R$. Given history $\theta = (y^0, u^0, \ldots, y^{n-1}, u^{n-1}, y^n)$, we define the notation $\theta(k) := (y^0, u^0, \ldots, y^{k-1}, u^{k-1}, y^k)$ and $tgt(\theta) := y^n$, as was the case with runs. □

Definition III.7 (Supervisor)**.** A supervisor σ for a system $S = (X, U, \rightarrow, Y, H)$ is a function $\sigma : \Theta \rightarrow 2^U$ which chooses which control inputs to enable/disable after each history. A supervisor is called memoryless if it is of the form $\sigma : Y \rightarrow 2^U$. A run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n) \in R_n(S)$ is allowed by supervisor σ if $u^i \in \sigma(\theta(\rho(i)))$, for $i = 0, \ldots, n-1$. □

Definition III.8 (Specification)**.** A safety specification for a system $S = (X, U, \rightarrow, Y, H)$ is a subset $Safe \subseteq \rightarrow$ of transitions that we would like the system S to be restricted to. A marking specification for S is a set $X_m \subseteq X$ of "special" or marked states. We say that S is deadlocking if there exists a run ρ such that $U(tgt(\rho)) = \emptyset$ and $tgt(\rho) \notin X_m$. □

In discrete event systems, marked states are used to denote states where some operation of interest has completed.

3.6.2 The State Reduction

Definition III.9 (State Reduction)**.** Given two systems S_a and S_b with $Y_a = Y_b = Y$, we say that S_a is a state reduction of S_b with state relation $R \subseteq X_a \times X_b$ and output dependent control relation $C : Y \rightarrow 2^{U_a \times U_b}$ (hereafter referred to only as control relation) if:

1. $R^{-1} = \{(x_b, x_a) \subseteq X_b \times X_a : (x_a, x_b) \in R\}$ is a function.

2. For every $y \in Y$, the relation $C(y) \subseteq U_a \times U_b$ is a bijection relation.

3. $H_a(x_a) = H_b(x_b)$ if and only if $(x_a, x_b) \in R$.

4. $\forall (x_a, u_a, x'_a) \in \rightarrow_a$, $\exists (x_b, u_b, x'_b) \in \rightarrow_b$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(H_a(x_a))$, and $(x'_a, x'_b) \in R$.

5. $\forall (x_b, u_b, x'_b) \in \rightarrow_b$, $\exists (x_a, u_a, x'_a) \in \rightarrow_a$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(H_b(x_b))$, and $(x'_a, x'_b) \in R$.

□

Remark III.10. The state reduction was first defined in (*Dallal et al.*, 2013b), where we used slightly different conditions. In this work, we have changed notation for the control relation C to resolve ambiguity. Furthermore, condition 5) was previously stated as: $\forall (x_a, x_b) \in R$, $(u_a, u_b) \in C$ and $x'_b \in \mathbf{Post}_{u_b}(x_b)$, $\exists x'_a \in \mathbf{Post}_{u_a}(x_a)$ such that $(x'_a, x'_b) \in R$. The two conditions can be shown to be equivalent under conditions 1) and 2). □

In words, condition 1) signifies that every $x_b \in X_b$ is in relation with exactly one $x_a \in X_a$, condition 5) signifies that, for every $(x_b, u_b, x'_b) \in \rightarrow_b$, there exists $(x_a, u_a, x'_a) \in \rightarrow_a$ which models $(x_b, u_b, x'_b) \in \rightarrow_b$, and condition 4) signifies that every transition in $\rightarrow_a$ models *some* transition in $\rightarrow_b$. Significantly, conditions 4) and 5) can be achieved by construction for any system S_b, and relations R and C satisfying conditions 1), 2), and 3). Furthermore, the system S_a is uniquely defined by S_b, R, and C.

Definition III.11 (Induced Specification)**.** Given system S_b with state reduction S_a, along with safety and marking specifications $Safe_b \subseteq \rightarrow_b$ and $X_{m,b} \subseteq X_b$ on system

S_b, define the induced specification on S_a as follows:

$$
\begin{aligned}
& (x_a, u_a, x_a') \in Safe_a \subseteq \rightarrow_a \\
\Leftrightarrow \quad & \left\{ \begin{array}{l} (x_b, u_b, x_b') \in \rightarrow_b \text{ s.t. } (x_a, x_b) \in R \\ \quad \wedge (u_a, u_b) \in C(H_a(x_a)) \wedge (x_a', x_b') \in R \end{array} \right\} \subseteq Safe_b \qquad (3.7) \\
& X_a \in X_{m,a} \subseteq X_a \Leftrightarrow \{x_b \in X_b \text{ s.t. } (x_a, x_b) \in R\} \subseteq X_{m,b} \qquad (3.8)
\end{aligned}
$$

In the remainder of this work we will often refer to the computation of a maximally permissive, safe, and non-blocking supervisor of a system. We will leave the discussion of its computation and of the translation between DES in the form of Def. III.2 and systems in the form of Def. III.4 to the next section.

The usefulness of Def. III.9 is illustrated in the following theorem:

Theorem III.12. *Suppose that system S_a is a state reduction of system S_b with state relation R and control relation C and that we are given safety and marking specifications $Safe_b \subseteq \rightarrow_b$ and $X_{m,b} \subseteq X_b$ for system S_b. Let $Safe_a$ and $X_{m,a}$ be the corresponding induced specifications for system S_a and suppose that we have a maximally permissive, safe, and non-deadlocking supervisor $\sigma_a : Y \rightarrow 2^{U_a}$, where Y is the (common) output space. Define the supervisor $\sigma_b : Y \rightarrow 2^{U_b}$ by $u_b \in \sigma_b(y)$ iff $\exists u_a \in \sigma_a(y)$ such that $(u_a, u_b) \in C(y)$. Then σ_b is safe, non-deadlocking, and maximally permissive among supervisors of the form $\sigma_b : Y \rightarrow 2^{U_b}$.*

Proof. We proceed in three claims. The first two claims show that the supervisor σ_b will be safe and non-deadlocking. To prove maximal permissiveness, we first define a function $\sigma_{b \rightarrow a}$ which maps supervisors for S_b to supervisors for S_a such that $\sigma_a = \sigma_{b \rightarrow a}(\sigma_b)$, for σ_a and σ_b as defined in the theorem statement. The last claim shows that, for any σ_b' that is safe an non-deadlocking for S_b, $\sigma_a' = \sigma_{b \rightarrow a}(\sigma_b')$ will be safe and non-deadlocking for S_a. Maximal permissiveness then follows by a monotonicity

property on $\sigma_{b\to a}$.

Claim 1: $(x_a, x_b) \in R \wedge (\sigma_a(H_a(x_a)) \neq \emptyset \vee x_a \in X_{m,a}) \Rightarrow (\sigma_b(H_b(x_b)) \neq \emptyset \vee x_b \in X_{m,b})$. By definition of $X_{m,a}$, $x_a \in X_{m,a} \Leftrightarrow x_b \in X_{m,b}$, for all $x_b : (x_a, x_b) \in R$, so that $(x_a, x_b) \in R \wedge x_a \in X_{m,a} \Rightarrow x_b \in X_{m,b}$. By definition of σ_b, $u_b \in \sigma_b(y)$ iff $\exists u_a \in \sigma_a(y)$ such that $(u_a, u_b) \in C(y)$. But $(x_a, x_b) \in R \Rightarrow H_a(x_a) = H_b(x_b)$, so that $(x_a, x_b) \in R \wedge \sigma_a(H_a(x_a)) \neq \emptyset \Rightarrow \sigma_b(H_b(x_b)) \neq \emptyset$.

By assumption, $H_b(x_{b,1}) = H_b(x_{b,2}) \Rightarrow U_b(x_{b,1}) = U_b(x_{b,2})$, from which it follows that, if $\sigma_b(H_b(x_b)) \neq \emptyset$, then x_b is not deadlocked under σ_b.

Claim 2: $\forall x_b \in X_b,\ u_b \in \sigma_b(H_b(x_b)) \Rightarrow \forall x'_b \in \mathbf{Post}_{u_b}(x_b), (x_b, u_b, x'_b) \in Safe_b \wedge [\sigma_b(H_b(x'_b)) \neq \emptyset \vee x'_b \in X_{m,b}]$.

Consider any $x_b \in X_b$ and any $u_b \in \sigma_b(H_b(x_b))$. By property (1) of Def. III.9, R^{-1} is a function. Therefore let x_a be the unique member of X_a such that $(x_a, x_b) \in R$. By property (3) of Def. III.9, $H_a(x_a) = H_b(x_b) = y$ for some $y \in Y$. By property (2) of Def. III.9, $C(y)$ is a bijection. Therefore let u_a be the unique member of U_a such that $(u_a, u_b) \in C(y)$. From the definition of σ_b, it follows that $u_a \in \sigma_a(H_a(x_a))$. Thus, $\forall x'_a \in \mathbf{Post}_{u_a}(x_a), (x_a, u_a, x'_a) \in Safe_a \wedge [\sigma_a(H_a(x'_a)) \neq \emptyset \vee x'_a \in X_{m,a}]$. From the way that $Safe_a$ was defined, this implies that $(x_b, u_b, x'_b) \in Safe_b$, for all $x'_b \in \mathbf{Post}_{u_b}(x_b)$ such that $(x'_a, x'_b) \in R$ and $x'_a \in \mathbf{Post}_{u_a}(x_a)$. But property (5) of Def. III.9 states that $\forall x'_b \in \mathbf{Post}_{u_b}(x_b)$, $\exists x'_a \in \mathbf{Post}_{u_a}(x_a)$ satisfying $(x'_a, x'_b) \in R$. From this and the previous statement, it follows that $(x_b, u_b, x'_b) \in Safe_b$, for all $x'_b \in \mathbf{Post}_{u_b}(x_b)$. It similarly follows from property (5) of Def. III.9 along with Claim 1 that $\sigma_b(H_b(x'_b)) \neq \emptyset \vee x'_b \in X_{m,b}$, for all $x'_b \in \mathbf{Post}_{u_b}(x_b)$.

Thus σ_b is safe and non-deadlocking. Given any supervisor $\sigma'_b : Y \to 2^{U_b}$, let $\sigma'_a : Y \to 2^{U_a}$ be defined by $u_a \in \sigma'_a(y)$ iff $\exists u_b \in \sigma'_b(y)$ such that $(u_a, u_b) \in C(y)$ and let the function $\sigma_{b\to a}$ be the mapping which takes a supervisor σ'_b for system b to the supervisor σ'_a for system a in this way.

Claim 3: If σ'_b is safe and non-deadlocking then so is $\sigma'_a = \sigma_{b\to a}(\sigma'_b)$.

Suppose that σ'_b is safe and non-deadlocking and take $\sigma'_a = \sigma_{b\to a}(\sigma'_b)$. Consider any $x_a \in X_a$, any $u_a \in \sigma'_a(H_a(x_a))$, and any $x'_a \in \mathbf{Post}_{u_a}(x_a)$. By property (4) of Def. III.9, there exists $(x_{b,1}, u_b, x'_{b,1}) \in \to_b$ such that $(x_a, x_{b,1}) \in R$, $(u_a, u_b) \in C$, and $(x'_a, x'_{b,1}) \in R$. From the definition of $\sigma_{b\to a}(\sigma'_b)$, it must be that $u_b \in \sigma'_b(H_b(x_{b,1}))$. By property (3) of Def. III.9, any $x_{b,2}$ such that $(x_a, x_{b,2}) \in R$ must satisfy $H_b(x_{b,1}) = H_a = H_b(x_{b,2})$. It follows that $u_b \in \sigma'_b(H_b(x_b))$, for *any* $x_b \in X_b$ such that $(x_a, x_b) \in R$. Since σ'_b is safe and non-deadlocking, it follows that, for all $(x_b, u_b, x'_b) \in \to_b$ such that $(x_a, x_b) \in R$, we have that $(x_b, u_b, x'_b) \in Safe_b \wedge [\sigma'_b(H_b(x'_b)) \neq \emptyset \vee x'_b \in X_{m,b}]$. It follows from the definition of $Safe_a$ that $(x_a, u_a, x'_a) \in Safe_a$ and from the definitions of $\sigma_{b\to a}(\sigma'_b)$ and $X_{m,a}$ that $\sigma'_a(H_a(x'_a)) \neq \emptyset \vee x'_a \in X_{m,a}$. Since $x'_a \in \mathbf{Post}_{u_a}(x_a)$ was arbitrary, it follows that u_a is a safe and non-deadlocking control decision from x_a. Since $x_a \in X_a$ and $u_a \in \sigma'_a(H_a(x_a))$ were arbitrary, it follows that $\sigma'_a = \sigma_{b\to a}(\sigma'_b)$ is safe and non-deadlocking.

It is obvious from the definition of $\sigma_{b\to a}$ that $\sigma'_b \supseteq \sigma_b \Rightarrow \sigma_{b\to a}(\sigma'_b) \supseteq \sigma_{b\to a}(\sigma_b) = \sigma_a$. Thus, if there exists a safe and non-deadlocking supervisor $\sigma'_b \supseteq \sigma_b$ then it follows that σ_a is not maximally permissive, a contradiction. □

The above theorem shows that it is possible to compute a supervisor for a system with a large or infinite state space by abstracting that system to one with a finite state space, computing a supervisor for the reduced system, and translating back. Furthermore, this process conserves not only safety and non-deadlockingness in the translation, but also maximal permissiveness.

Remark III.13. The above theorem characterizes a controller σ as safe and non-deadlocking for system $S = (X, U, \to, Y, H)$, safety specification $Safe$, and marking specification X_m if and only if $\forall x \in X$, $\forall u \in \sigma(H(x))$, $\forall x' \in \mathbf{Post}_u(x)$, we have that $(x, u, x') \in Safe \wedge (\sigma(H(x')) \neq \emptyset \vee x' \in X_m)$. This is a sufficient condition for a system to be safe and non-deadlocking, but it is not necessary if the supervisor can use initial state information, even if we restrict attention to memoryless supervisors.

For an example of such a situation, see Example III.14. □

Example III.14. Figure 3.4 shows an example of a system (left) and its corresponding state reduction (right). If we assume that there is only a marking specification and no safety specification, then the maximally permissive supervisor σ_1 for the state reduction would enable $\{a, b\}$ from state $\{1, 2\}$, $\{a\}$ from state $\{3, 4, 5\}$, and $\{a, b, c\}$ from state $\{7\}$. It can be seen that this would indeed be a maximally permissive memoryless solution for the left system if there were no initial state information. If, however, the initial state is known a priori to be one of $\{1, 2\}$, then there exists a strictly more permissive memoryless supervisor σ_2 for the left system which also enables b from states $\{3, 4, 5\}$. It is possible to be more permissive from states $\{3, 4, 5\}$ by making use of the fact that the initial states are $\{1, 2\}$ and event c was disabled from states $\{1, 2\}$, making state 5 unreachable. Another safe memoryless supervisor σ_3 enables $\{a\}$ from states $\{1, 2\}$, $\{a, b, c\}$ from states $\{3, 4, 5\}$ and $\{a, b, c\}$ from states $\{7\}$. Thus, it is possible to enable more from states $\{3, 4, 5\}$ by enabling less from states $\{1, 2\}$. Consistent with the discussion of Remark III.13, both of these supervisors violate the property that $\forall x \in X$, $\forall u \in \sigma(H(x))$, $\forall x' \in \mathbf{Post}_u(x)$, we have that $(x, u, x') \in Safe \wedge (\sigma(H(x')) \neq \emptyset \vee x' \in X_m)$. In particular, σ_2 and σ_3 both allow b from state 5, despite the fact that this allows $(5, b, 8)$, and state 8 is deadlocked. Furthermore, the union of σ_2 and σ_3 is blocking, since it allows the string bc, which leads to blocking state 8. Thus, there does not exist a maximally permissive safe and non-blocking supervisor which uses the initial state information in this case. Note that the system on the left is accessible, deterministic, and has both initial and marked states which respect the partition of states determined by the output map. This example is very closely related to the problem of obtaining maximally permissive supervisors of the form $S : X_G \to 2^E$ for a discrete event system G, subject to specification automaton H, which would normally require the supervisor to be defined over the state space of the product automaton $G \times H$. □

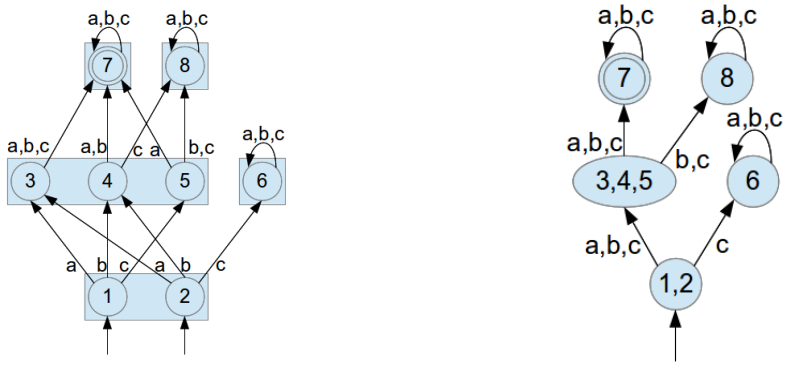


Figure 3.4: A system and its corresponding state reduction. States of the left system with the same output are placed in a common box. We use the usual DES convention of denoting marked states with a double circle and initial states with an incoming arrow that has no source state.

3.6.3 The Exact State Reduction

Definition III.15 (Exact State Reduction (2))**.** Given two systems S_a and S_b with $Y_a = Y_b = Y$, we say that S_a is an exact state reduction of S_b with state relation $R \subseteq X_a \times X_b$ and control relation $C : Y \to 2^{U_a \times U_b}$ if S_a is a state reduction of S_b with state and control relations R and C and:

6. $\forall (x_a, u_a, x'_a) \in \rightarrow_a, \forall x'_b \in X_b : (x'_a, x'_b) \in R, \exists (x_b, u_b, x'_b) \in \rightarrow_b$ such that $(x_a, x_b) \in R$ and $(u_a, u_b) \in C(H_a(x_a))$. □

The above condition is akin to a time-reversed alternating similarity condition, in the sense that it requires that every transition of S_a have a corresponding transition in S_b, for every pair of related *target* states, rather than for every pair of related *source* states. Lemma III.17 demonstrates its usefulness.

Remark III.16. The exact state reduction was first defined in (*Dallal et al.*, 2013b), where we used a normal (i.e., non time-reversed) alternating similarity condition. We have added the "(2)" in this work to differentiate between these. □

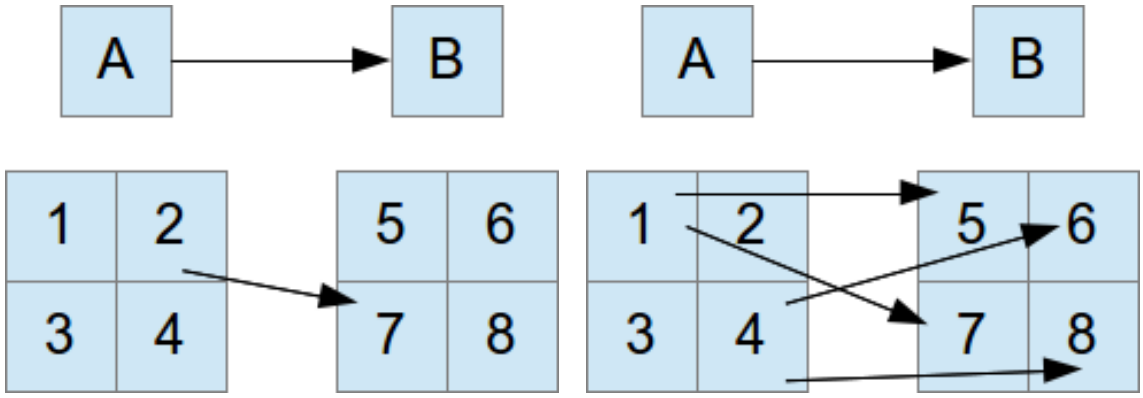
Lemma III.17. *Suppose that system S_b has an exact state reduction (2) S_a. Then, for any history θ_b for system S_b and any $x_b \in X_b$ such that $H(x_b) = tgt(\theta_b)$, there exists a run ρ_b such that $\theta_b = \theta(\rho_b)$ and $x_b = tgt(\rho_b)$.*

Proof. The proof is by induction on the length of θ_b. The base case is trivially true. Assume that the lemma holds up to histories of length n and consider a pair of histories $\theta_b \in \Theta_n(S_b)$ and $\theta'_b \in \Theta_{n+1}(S_b)$ such that θ_b is a prefix of θ'_b. Also define $y = tgt(\theta_b)$, $y' = tgt(\theta'_b)$, and let $\rho'_b = (x_b^0, \ldots, x_b^n, u_b^n, x_b^{n+1}) \in R_{n+1}(S_b)$ be such that $\theta'_b = \theta(\rho'_b)$. Note that, in particular, this implies $H_b(x_b^n) = y$ and $H_b(x_b^{n+1}) = y'$. Since $(x_b^n, u_b^n, x_b^{n+1}) \in \rightarrow_b$, we have from property (5) that $\exists (x_a^n, u_a^n, x_a^{n+1}) \in \rightarrow_a$ such that $(x_a^n, x_b^n) \in R$, $(u_a^n, u_b^n) \in C(H_b(x_b^n)) = C(y)$ and $(x_a^{n+1}, x_b^{n+1}) \in R$. From property (3), we have $H_a(x_a^n) = H_b(x_b^n) = y$ and $H_a(x_a^{n+1}) = H_b(x_b^{n+1}) = y'$. Now consider any $x'_b \in X_b$ such that $H(x'_b) = tgt(\theta'_b) = y'$. Using property (3) again, we have that $(x_a^{n+1}, x'_b) \in R$. From property (6) we therefore have that $\exists (x_b, u_b, x'_b) \in \rightarrow_b$ such that $(x_a^n, x_b) \in R$ and $(u_a^n, u_b) \in C(H_a(x_a^n)) = C(y)$. From property (3), we have that $H_b(x_b) = H_a(x_a^n) = y$ and from property (2) we have that $u_b = u_b^n$. From the induction hypothesis, there exists a run ρ_b such that $\theta_b = \theta(\rho_b)$ and $tgt(\rho_b) = x_b$. Thus we can form the run $\rho''_b := \rho_b.u_b.x'_b$ satisfying $\theta'_b = \theta(\rho''_b)$ and $tgt(\rho''_b) = x'_b$, which completes the proof. □

In words, the above lemma implies that, when there exists an exact state reduction (2) for system S_b, a history θ_b gives no more information about the current state of S_b than does the last output $tgt(\theta_b)$. The following theorem follows immediately from this observation.

Theorem III.18. *Suppose that system S_a is an exact state reduction (2) of system S_b and that all other conditions of Thm. III.12 are satisfied. Then the obtained supervisor σ_b will be safe, non-deadlocking, and maximally permissive among supervisors of the form $\sigma_b : \Theta \rightarrow 2^{U_b}$.*

Remark III.19. As in the case of (non-exact) state reductions, the obtained supervisor will not generally be maximally permissive if the supervisor can use initial state information. In particular, if the set of initial states $X_{0,b}$ gives more information



(a) A State Reduction (b) An Exact State Reduction

Figure 3.5: A depiction of the state reduction (left) and exact state reduction (right) for a simple system $S_b = (\{1, \ldots, 8\}, \{u\}, \rightarrow_b, \{A, B\}, H_b)$, where $H_b(x) = A$ for $x \in \{1, \ldots, 4\}$ and $H_b(x) = B$ for $x \in \{5, \ldots, 8\}$. In both the left and right cases, there is a transition $(x, u, x') \in \rightarrow_b$ with $x \in H_b^{-1}(A)$ and $x' \in H_b^{-1}(B)$, and hence a transition from A to B in the corresponding state reduction. The system on the right contains some transition $(x, u, x') \in \rightarrow_b$ with $x \in H_b^{-1}(A)$, *for every* $x' \in H_b^{-1}(B)$. For the system on the left, the occurrence of a transition from A to B in the state reduction allows us to determine that S_b is in state 7. For the system on the right, this transition only allows to determine that the system is some state in the set $H_b^{-1}(B)$.

than the initial output y_0, then there may exist more permissive supervisors. Note however that, if $H(x_{b,1}) = H(x_{b,2}) \Rightarrow [x_{b,1} \in X_{0,b} \Leftrightarrow x_{b,2} \in X_{0,b}]$, then $X_{0,b}$ gives no more information than the initial output y_0, and hence the resulting supervisor will still be maximally permissive. This is contrary to the case of non-exact state reductions, in which case the above condition is still not sufficient to guarantee maximal permissiveness of the supervisor σ_2 obtained in Thm. III.12, as is demonstrated in Ex. III.14.

Figure 3.5 depicts an example of a state reduction and an example of an exact state reduction.

3.7 Supervisor Computation and Relations Between the Time-discretized and Discrete Event Systems

In this section, we describe the supervisory control problem of DES, prove that G is a state reduction of the time-discretized system of equation 3.2, and use this to show how we can solve Prob. III.1. Readers familiar with supervisory control theory of DES may skip Subsec. 3.7.1.

3.7.1 Supervisory Control Theory of DES

Given a set of events E, E^* denotes the set of finite strings of events in E. A set of strings $K \subseteq E^*$ is called a *language*. The *prefix-closure* of a language $K \subseteq E^*$, denoted by $\overline{K}$, is defined by $\overline{K} = \{s \in E^* : \exists t \in E^* \wedge st \in K\}$. Recall that a deterministic DES G is defined as a tuple $G = (X, E, f, x_0, X_m)$, where X is a set of states, E is a set of events, $f : X \times E \to X$ is a (partial) transition function, $x_0 \in X$ is the initial state, and $X_m \subseteq X$ is a set of marked or accepting states. The transition function f is extended from events to strings through $f(x, se) = f(f(x, s), e)$. The language generated by G, denoted by $\mathcal{L}(G)$, is defined as $\mathcal{L}(G) := \{s \in E^* : f(x_0, s)!\}$, where ! means "is defined". The marked language of G, denoted by $\mathcal{L}_m(G) \subseteq \mathcal{L}(G)$ is defined by $\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) : f(x_0, s) \in X_m\}$.

Obtaining a safe, non-blocking, and maximally permissive supervisor for a discrete event system consists of solving the basic supervisory control problem in the non-blocking case, or BSCP-NB, as described in (*Ramadge and Wonham*, 1987), (*Cassandras and Lafortune*, 2008). Specifically, problem BSCP-NB computes the supremal controllable sublanguage of a specification $\mathcal{L}_m(H)$ with respect to $\mathcal{L}(G)$, where G is a system automaton and H is a specification automaton satisfying $\mathcal{L}(H) \subseteq \mathcal{L}(G)$ and $\mathcal{L}_m(H) \subseteq \mathcal{L}_m(G)$. The set $\mathcal{L}(H)$ represents the legal sublanguage of $\mathcal{L}(G)$, representing safe system behavior. The set $\mathcal{L}_m(H)$ is usually assumed to satisfy the

property $\mathcal{L}_m(H) = \mathcal{L}(H) \cap \mathcal{L}_m(G)$ (called $\mathcal{L}_m(G)$-closure).

In general, the event set of G and H, denoted by E is partitioned into controllable events E_c and uncontrollable events E_{uc}. The solution to problem BSCP-NB is the language $(\mathcal{L}_m(H))^{\uparrow C}$, where $\uparrow C$ denotes the supremal controllable sublanguage operation. This is the largest sublanguage $K \subseteq \mathcal{L}_m(H)$ satisfying the property $\overline{K}E_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$, which means that there exist no strings in $\overline{K}$ that can be extended by an uncontrollable event to a string in $\mathcal{L}(G) \setminus \overline{K}$. The standard algorithm which solves this problem is given in (*Wonham and Ramadge*, 1987) and constructs a supervisor S such that $\mathcal{L}_m(S/G) = (\mathcal{L}_m(H))^{\uparrow C}$ and $\mathcal{L}(S/G) = \overline{(\mathcal{L}_m(H))^{\uparrow C}}$, where S/G is the system G controlled by S.

3.7.2 Translating Between Transition Systems and Discrete Event Systems

In Thm. III.12, we compute the maximally permissive, safe, and non-deadlocking supervisor of a system $S_a = (X_a, U_a, \rightarrow_a, Y_a, H_a)$ with respect to a safety specification $Safe_a \subseteq \rightarrow_a$ and set of marked states $X_{m,a} \subseteq X_a$. In practice, this would be done by translating the transition system S_a into an automaton G_a, the specification into an automaton H_a, and solving BSCP-NB to obtain the supervisor S described in the preceding subsection. Consistent with the above description of BSCP-NB, we need two automata, denoted by G_a and H_a, to capture the system behavior $\rightarrow_a$ and the legal behavior given by $Safe_a$ and $X_{m,a}$. The automaton $G_a := (X_a \cup Z_a, E_c \cup E_{uc}, \psi_{G_a}, x_{a,0}, X_{m,a})$ must satisfy the following conditions:

$$E_c = U_a \tag{3.9}$$

$$\psi_{G_a} \subseteq (X_a \times E_c \times Z_a) \cup (Z_a \times E_{uc} \times (X_a \cup Z_a)) \tag{3.10}$$

$$\psi_{G_a}(x_a, u_a)! \Leftrightarrow \exists x'_a \in X_a : (x_a, u_a, x'_a) \in \rightarrow_a \tag{3.11}$$

$$\exists t \in E_{uc}^* : \psi_{G_a}(x_a, u_a t) = x'_a \Leftrightarrow (x_a, u_a, x'_a) \in \rightarrow_a, \tag{3.12}$$

where Z_a is some set of intermediate states (such as Q_{I1} and Q_{I2} of DES G in Sec. 3.5). In words, Eq. (3.9) signifies that the controllable events of G_a are the control inputs of S_a, whereas Eq. (3.10) signifies that controllable (resp. uncontrollable) events are defined only from states in X_a (resp. Z_a) and lead only to states in Z_a (resp. $X_a \cup Z_a$). Eqs. (3.11) and (3.12) signify that, for every $(x_a, u_a, x'_a) \in \rightarrow_a$, event u_a is defined from state x_a of G_a and there exists some uncontrollable sequence of events following u_a that takes G_a from $\psi_{G_a}(x_a, u_a)$ to x'_a. Thus, we use uncontrollable events to model any non-determinism in the transition relation $\rightarrow_a$. Thus, there is a true 1-1 equivalence between a system in the form of definition III.4 and a discrete-event system in the form above. Given G_a, $Safe_a$ and $X_{m,a}$, we construct subautomaton $H_a \sqsubseteq G_a$ (G_1 is a subautomaton of G_2 if $\psi_1(x_{01}, s) = \psi_2(x_{02}, s)$ for all $s \in \mathcal{L}(G_1)$, where x_{01} and x_{02} are the initial states of G_1 and G_2, see (*Cassandras and Lafortune*, 2008)) such that $X_{m,a} \subseteq X_a$ is the set of marked states and with transition function ψ_{H_a} satisfying conditions (3.11) and (3.12), but with $Safe_a$ instead of $\rightarrow_a$. We solve BSCP-NB for H_a and G_a. Because $H_a \sqsubseteq G_a$, S is of the form $S : X_a \rightarrow 2^{U_c}$.

To unify notation between systems as in Def. III.4 and discrete event systems as described above, we will use the notation $U(x) := \{u \in E_c : \psi(x, u)!\}$ and $\mathbf{Post}_u(x) := \{x' \in X_a : (\exists t \in E_{uc}^*)(x' = \psi(x, ut))\}$ for $x \in X_a$ and (in an abuse of notation) will write $(x, u, x') \in \psi$ if $x \in X_a$ and $x' \in \mathbf{Post}_u(x)$. This notation allows us to work with DES of the above form in the context of state reductions and exact state reductions.

3.7.3 Relations Between the Time-discretized and Discrete Event Systems

Proposition III.20. *Define the observation maps $H_{\tilde{Q}}(q) := q$, $H_X(x) := \ell(x)$, the relation $R := \{(q, x) \in \tilde{Q} \times X : \ell(x) = q\}$, and the control relation $C(q) := \{(u_c, v_c) : v_c\tau = u_c \in U_c\}$, for all $q \in \tilde{Q}$. Then DES G of Sec. 3.5 is a state reduction of system* (3.2).

Proof. Properties (1), (2), and (3) follow immediately from the definitions of H_X, H_Q, ℓ, R, and C.

Property (4): Consider any $q \in \tilde{Q}$, $u_c \in U_c$, $u_{uc} \in U_{uc}$, and $w \in W$, with $q' = \psi(q, u_c, u_{uc}, w) = q + u + w$ (where $u = (u_c, u_{uc})$). We construct $x \in X$, $x' \in X$ and $\delta \in \Delta$ such that $\ell(x) = q$, $\ell(x') = q'$, and $x + u + \delta = x'$ by considering each co-ordinate in turn. There are three cases, depending on where w_i lies with respect to the interval $[\delta_{\min}, \delta_{\max}]$ (recall from Eq. (3.4) that w_i may be smaller than $\delta_{\min}$ or larger than $\delta_{\max}$ when these values are not integer multiples of $\mu\tau$, because of the floor and ceiling operations).

Case 1: $\delta_{min} \leq w_i \leq \delta_{max}$**.** Take $x_i = q_i$, $\delta_i = w_i$, and $x'_i = q'_i$.

Case 2: $w_i > \delta_{max}$**.** Take $x_i = q_i + \mu\tau/2$, $\delta_i = \delta_{max}$, and $x'_i = x_i + u_i + \delta_{max}$. From the definition of ℓ, we have that $\ell_i(x_i) = q_i$. With these values, we obtain $q'_i - x'_i = q_i - x_i + w_i - \delta_{max} = w_i - \delta_{max} - \mu\tau/2$. From the definition of W, we know that $\delta_{max} < w_i < \delta_{max} + \mu\tau$, or equivalently that $0 < w_i - \delta_{max} < \mu\tau$. From this and the previous statement, we obtain $-\mu\tau/2 < q'_i - x'_i < \mu\tau/2$, from which it follows that $\ell(x'_i) = q'_i$.

Case 3: $w_i < \delta_{min}$**.** Take $x'_i = q'_i + \mu\tau/2$, $\delta_i = \delta_{min}$, and $x_i = x'_i - u_i - \delta_{min}$. The same reasoning as in the previous case shows that $\ell(x) = q$ and that $\ell(x') = q'$.

Property (5): Consider any $x \in X$, $u_c \in U_c$, $u_{uc} \in U_{uc}$, and $\delta \in \Delta$, with $x' = x + u + \delta$ (where $u = (u_c, u_{uc})$). Take $q = \ell(x)$, $q' = \ell(x')$, and $w = q' - q - u$. It suffices to show that $w \in W$. From $q = \ell(x)$ and $q' = \ell(x')$, we have $-\mu\tau/2 < x - q \leq \mu\tau/2$ and $-\mu\tau/2 < x' - q' \leq \mu\tau/2$ (component-wise). Combining these inequalities with $w = q' - q - u$ and $x' = x + u + \delta$, we obtain $w = \delta + (x - q) - (x' - q')$ and hence:

$$-\tau\mu + \delta < w < \delta + \tau\mu.$$

It follows that w is a vector whose components are all integer multiples of $\tau\mu$ and in the interval $(\delta_{min} - \mu\tau, \delta_{max} + \mu\tau)$. But from Eq. 3.4, this set of vectors is precisely equal to W, proving that $w \in W$. □

Proposition III.21. *Define $H_X(\cdot)$, $H_{\tilde{Q}}(\cdot)$, R, and C as in Prop. III.20. If δ_{min} and δ_{max} are both integer multiples of $\tau\mu$, then DES G of Sec. 3.5 is an exact state reduction (2) of system* (3.2).

Proof. Property (6): Consider any $q \in \tilde{Q}$, $u_c \in U_c$, $u_{uc} \in U_{uc}$, $w \in W$, and x' such that $q' = q + u + w = \ell(x')$, where $u = (u_c, u_{uc})$. We construct $x \in X$ and $\delta \in \Delta$ such that $q = \ell(x)$ and $x' = x + u + \delta$. Simply take $\delta = w$ and $x = x' - u - \delta$. As remarked in the proof of Prop. III.20, w must be a vector whose components are integer multiples of $\tau\mu$ and in the interval $(\delta_{min} - \mu\tau, \delta_{max} + \mu\tau)$. If δ_{min} and δ_{max} are multiples of $\tau\mu$, then it follows that the components of w are in the (closed) interval $[\delta_{min}, \delta_{max}]$. Thus $\delta \in \Delta$. Furthermore, $x' - x = q' - q = u + w$, so that $x - q = x' - q'$, from which it follows that $q' = \ell(x') \Rightarrow q = \ell(x)$. □

Given the above results, we can solve Prob. III.1.

Theorem III.22. *Define the automaton $H := (Q, E_Q \cup U_c \cup U_{uc} \cup W, \psi_{safe}, q_0, Q_m) \sqsubseteq G$, where $\psi_{safe} \subseteq \psi$ is defined by:*

$$(q, u_c, q') \in \psi_{safe} \Leftrightarrow \begin{array}{l} x \in \ell^{-1}(q) \wedge v_{uc} \in V_{uc} \wedge d \in D^{[0,\tau]} \wedge x' \in \ell^{-1}(q') \\ \wedge x' = x + v\tau + \int_0^\tau d(t)dt \\ \Rightarrow x(t) = x + vt + \int_0^t d(t')dt' \notin B,\ \forall t \in [0, \tau] \end{array}, \quad (3.13)$$

where $v = (u_c/\tau, v_{uc})$. Solve for the supremal controllable sublanguage $(\mathcal{L}_m(H))^{\uparrow C}$ of $\mathcal{L}_m(H)$ with respect to $\mathcal{L}(G)$ and uncontrollable event set $E_{uc} = U_{uc} \cup W$, obtaining a maximally permissive safe and non-blocking supervisor $S : \tilde{Q} \to 2^{U_c}$. Then the supervisor $\sigma : X/\ell \to 2^{V_c}$ defined by $v_c \in \sigma(x) \Leftrightarrow u_c = \tau v_c \in S(\ell(x))$ solves Prob. III.1.

Proof. Solving Prob. III.1 requires finding the maximally permissive safe and non-deadlocking supervisor σ for System $S_b = (X, V_c, \rightarrow_b, \tilde{Q}, \ell)$ subject to safety specification $Safe_b$ and marking $X_{m,b}$, where:

$$(x, v_c, x') \in \rightarrow_b \Leftrightarrow (\exists v_{uc} \in V_{uc})(\exists \delta \in \Delta) : x + \tau v + \delta = x', \quad v = (v_c, v_{uc}), \tag{3.14}$$

$$(x, v_c, x') \in Safe_b \Leftrightarrow \begin{array}{l} v_{uc} \in V_{uc} \wedge d \in D^{[0,\tau]} \\ \wedge x' = x + v\tau + \int_0^\tau d(t)dt \\ \Rightarrow x(t) = x + vt + \int_0^t d(t')dt' \notin B, \ \forall t \in [0, \tau] \end{array}, \tag{3.15}$$

and $X_{m,b} = \ell^{-1}(q_m)$. Thus, it suffices to apply Thm. III.12, and we proceed to verify its conditions. Proposition III.20 shows that G is a state reduction of S_b, with the state and control relations $R = \{(q, x) \in \tilde{Q} \times X : q = \ell(x)\}$ and $C(q) = \{(u_c, v_c) \in U_c \times V_c : u_c = v_c\tau\}$. The safety specification $Safe_a$ defined by equation (3.13) does indeed satisfy the condition $(q, u_c, q') \in Safe_a$ if and only if, for all $(x, v_c, x') \in \rightarrow_b$ such that $(q, x) \in R$, $(u_c, v_c) \in C(q)$ and $(q', x') \in R$, we have that $(x, v_c, x') \in Safe_b$. Finally, the set Q_m of marked states for G and H obviously satisfies the condition $q \in Q_m$ if and only if $x \in X_{m,b}$ for all $x \in X_b$ such that $(q, x) \in R$, since $Q_m = \{q_m\}$, $X_{m,b} = \ell^{-1}(q_m)$, and $(q, x) \in R \Leftrightarrow q = \ell(x)$. Thus, G is a state reduction of S_b, and $Safe_a$ and $X_{m,a} = Q_m$ are induced specifications, satisfying the conditions of Thm. III.12. □

Theorem III.23. *If δ_{min} and δ_{max} are both integer multiples of $\tau\mu$, then the supervisor σ of Thm. III.22 solves Prob. III.1, and is maximally permissive among the class of all supervisors, not merely memoryless ones.*

Proof. Immediate from Prop. III.21, Thm. III.18, and the proof of Thm. III.22. □

3.8 Algorithmic Implementation

In this section, we provide an algorithm for computing the DES supervisor S of Thm. III.22 that is based on a depth-first search (DFS) and has a lower asymptotic complexity than the standard algorithm. The customized algorithm of this section is based on the following three observations: the vehicle's velocities are bounded by $\mu > 0$; the specification automaton H is a sub-automaton of G; and each pair of events $u_{uc}w \in U_{uc}W$ is feasible after each event U_c from each state $q \in \tilde{Q}$. The first observation implies that the system is acyclic, and hence livelock-free. This allows for solving problem BSCP-NB in time linear in the size of $G \times H$, rather than quadratic (see, e.g. (*Hadj-Alouane et al.*, 1994)). The second observation implies that the product automaton $H \times G$ is isomorphic to H which, combined with the first observation, allows for the problem to be solved through a DFS on G. Finally, the third observation implies that there is no need to determine the safety of each string $u_c u_{uc} w \in U_c U_{uc} W$ from each state q. Instead, a single test of safety for each $u_c \in U_c$ and state $q \in \tilde{Q}$ suffices. This is formalized below.

Definition III.24. Given $q \in \tilde{Q}$, $u \in U$, $w \in W$, and $t \in [0, \tau]$, let the set $A_{q,u,w}(t) \subseteq X$ denote the set of points x_t such that there exist $x \in \ell^{-1}(q)$, $d : [0, \tau] \to D$, and $x' \in \ell^{-1}(\psi(q, u_c u_{uc} w))$ satisfying $x' = x + u + \int_0^\tau d(t')dt'$ and $x_t = x + u(t/\tau) + \int_0^t d(t')dt'$.

In words, $A_{q,u,w}(t)$ represents the set of possible vehicle positions x_t such that it is possible for $x(k\tau + t) = x_t$ when $\ell(x(k\tau)) = q$, DE control decision u_c is issued at time $k\tau$, the uncontrolled vehicles take action u_{uc}, and the disturbance event is w. Given the above definition, we can say that the transition corresponding to initial state $q \in \tilde{Q}$, action of the controlled vehicles u_c, action of the uncontrolled vehicles u_{uc} and disturbance event w is safe if and only if $A_{q,u,w}(t) \cap B = \emptyset$ for all $t \in [0, \tau]$.

Now remark that, since $u_{uc} \in U_{uc}$ and $w \in W$ cannot be chosen, control action u_c can only be allowed from state $q \in \tilde{Q}$ if $A_{q,u,w}(t) \cap B = \emptyset$ for all $u_{uc} \in U_{uc}$, $w \in W$, and

$t \in [0, \tau]$. Thus, we can define a second set $A_{q,u_c}(t)$ representing the set of possible vehicle positions x_t such that it is possible for $x(k\tau + t) = x_t$ when $\ell(x(k\tau)) = q$ and DE control decision u_c is issued at time $k\tau$, as is done below. In the following definition, we use $\mathbf{1} \in \mathbb{R}^n$ to denote the vector $(1, \ldots, 1)$, where n is the number of vehicles. For any two vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$, we write $a < b$ if $a_i < b_i$ for $i = 1, \ldots, n$, and similarly for $>$, $\leq$, and $\geq$. Finally, for $a, b \in \mathbb{R}^n$, let $(a, b]$ denote the box $\{c \in \mathbb{R}^n : a < c \leq b\}$.

Definition III.25. Given $t \in [0, \tau]$, let the set $A_{q,u_c}(t) \subseteq X$ denote the set of points x_t such that there exist $x \in \ell^{-1}(q)$, $u_{uc} \in U_{uc}$, and $d : [0, \tau] \to D$ satisfying $x_t = x + u(t/\tau) + \int_0^t d(t')dt'$, where $u = (u_c, u_{uc})$. This set is given by

$$A_{q,u_c}(t) = (q - \mathbf{1}\mu\tau/2 + \underline{v}_{u_c} t, q + \mathbf{1}\mu\tau/2 + \overline{v}_{u_c} t], \tag{3.16}$$

where:

$$\underline{v}_{u_c,i} = \begin{cases} u_{c,i}/\tau + d_{min}, & \text{vehicle } i \text{ is controlled} \\ v_{min} + d_{min}, & \text{vehicle } i \text{ is uncontrolled} \end{cases} \tag{3.17}$$

$$\overline{v}_{u_c,i} = \begin{cases} u_{c,i}/\tau + d_{max}, & \text{vehicle } i \text{ is controlled} \\ v_{max} + d_{max}, & \text{vehicle } i \text{ is uncontrolled} \end{cases} \tag{3.18}$$

From the above definitions, it follows that $A_{q,u_c}(t) = \bigcup_{u_{uc} \in U_{uc}, w \in W} A_{q,u,w}(t)$, and hence that the safety of control action u_c from state $q \in \tilde{Q}$ can be determined by checking if $A_{q,u_c}(t) \cap B = \emptyset$ for all $t \in [0, \tau]$. Equations for verifying this condition are give in the appendix. The general idea is to check intersection with the bad set for each pair of vehicles, so that the test takes $O(n^2)$ time.

The following proposition provides the basis for our algorithm:

Proposition III.26. *Let S be the supervisor that solves problem BSCP-NB for sys-*

tem automaton G of Sec. 3.5, specification automaton H of Thm. III.22, and event partition $E = E_c \cup E_{uc}$ with $E_c = E_Q \cup U_c$ and $E_{uc} = U_{uc} \cup W$. Then S is uniquely defined by:

$$\forall q \in \tilde{Q}, \left[u_c \in S(q) \Leftrightarrow \left(\begin{array}{c} A_{q,u_c}(t) \cap B = \emptyset \\ \wedge[(S(q') \neq \emptyset \vee q' = q_m),\ \forall q' \in \mathbf{Post}_{u_c}(q)] \end{array} \right) \right] \tag{3.19}$$

$$e_q \in S(q_0) \Leftrightarrow S(q) \neq \emptyset \tag{3.20}$$

Proof. First, $H \sqsubseteq G$, so that S does indeed have the form $S : Q \to 2^{U_c}$, rather than the more general form $S : \mathcal{L}(G) \to 2^{U_c}$. Furthermore, there are no controllable events defined from Q_{I1} or Q_{I2}, so what remains is only to define S for q_0 and $\tilde{Q}$. Consider any $q \in \tilde{Q}$. Comparing Eq. (3.13) and Def. III.25 shows that $[A_{q,u_c}(t) \cap B = \emptyset] \Leftrightarrow [(q, u_c, q') \in \psi_{safe},\ \forall q' \in \mathbf{Post}_{u_c}(q)]$. Thus allowing u_c from q satisfies the safety specification of H if and only if $A_{q,u_c}(t) \cap B = \emptyset$, from which it follows that $u_c \in S(q) \Rightarrow A_{q,u_c}(t) \cap B = \emptyset$. Furthermore, the condition $S(q') \neq \emptyset \vee q' = q_m$ means that q' is not deadlocked, from which it follows that $u_c \in S(q) \Rightarrow (S(q') \neq \emptyset \vee q' = q_m),\ \forall q' \in \mathbf{Post}_{u_c}(q)$. Because G is acyclic, non-blockingness is equivalent to non-deadlockness, which proves the $\Rightarrow$ portion of Eq. (3.19). The other direction follows from maximal permissiveness. A similar argument establishes the correctness of Eq. (3.20). Finally, S is uniquely defined by Equations (3.19) and (3.20) since G is acyclic, which implies that the recursive equations do not have cyclic dependencies. □

The above proposition provides the basis for the following algorithm.

It is assumed that, for every state $q \in \tilde{Q}$, there exists a boolean variable Done(q) that is initialized to false, and that $S(q)$ is initialized to the empty set.

Theorem III.27. *Algorithms 1 and 2 correctly compute the supervisor S of Prop. III.26.*

Algorithm 1 Computation of $S(q_0)$

```
1: procedure DoInit(G)
2:     S(q_0) ← E_Q
3:     for all e_q ∈ E_Q do
4:         if not DoDFS(G, q) then
5:             S(q_0) ← S(q) \ {e_q}
6:         end if
7:     end for
8: end procedure
```

Algorithm 2 DFS Computation of S

```
1: procedure DoDFS(G, q)
2:     if q = q_m then
3:         return true
4:     else if Done(q) then
5:         return [S(q) ≠ ∅]
6:     end if
7:     S(q) ← U_c
8:     for all u_c ∈ U_c do
9:         if [A_{q,u_c}(t) ∩ B ≠ ∅] then
10:            S(q) ← S(q) \ {u_c}
11:            continue
12:        end if
13:        for all q' ∈ Post_{u_c}(q) do
14:            if not DoDFS(G, q') then
15:                S(q) ← S(q) \ {u_c}
16:                break
17:            end if
18:        end for
19:    end for
20:    Done(q) ← true
21:    return [S(q) ≠ ∅]
22: end procedure
```

Proof. From lines 2-6 and 21, it follows that Algorithm 2 returns the boolean value of the expression $S(q) \neq \emptyset \vee q = q_m$. From this, Eq. (3.20), and Algorithm 1, it follows that $S(q_0)$ is correctly computed. From lines 7-19 of Algorithm 2, it follows that $u_c \in S(q) \Leftrightarrow [A_{q,u_c}(t) \cap B \neq \emptyset] \wedge [(S(q') \neq \emptyset \vee q' = q_m),\ \forall q' \in \mathbf{Post}_{u_c}(q)]$. It therefore follows from Eq. (3.19) that $S(q)$ is correctly computed. Finally, the DFS of Algorithm 2 ensures that the values will be correctly computed for all $q \in \tilde{Q}$ that are reachable under S. □

Proposition III.28. *The running time of Algorithms 1 and 2 is in* $O\left(|\tilde{Q}||U_c|\left[|\mathbf{Post}_{u_c}(q)| + n^2\right]\right)$.

Proof. Algorithm 2 is executed at most $|\tilde{Q}|$ times, once for each encountered $q \in \tilde{Q}$. The outer for loop (lines 8-19) is executed $|U_c|$ times and consists of verifying the condition $[A_{q,u_c}(t) \cap B \neq \emptyset]$ and executing the inner for loop. Verifying the condition $[A_{q,u_c}(t) \cap B \neq \emptyset]$ (line 9) takes $O(n^2)$ time (see Appendix). The inner for loop (lines 13-18) is executed $|\mathbf{Post}_{u_c}(q)|$ times, each of which takes $O(1)$ time beyond that of the recursive call. The total running time is therefore $O(|\tilde{Q}||U_c|\left[|\mathbf{Post}_{u_c}(q)| + n^2\right])$. □

Remark III.29. Because the particular state $q \in \tilde{Q}$ and control action $u_c \in U_c$ do not restrict the set of possible actions of the uncontrolled vehicles U_{uc} or the set of possible disturbance events W, the value $|\mathbf{Post}_{u_c}(q)|$ is independent of the particular $q \in \tilde{Q}$ and $u_c \in U_c$. This value is, however, dependent on the number of vectors of actions of the uncontrolled vehicles (which determines $|U_{uc}|$), as well as on the bounds of the disturbance (which determines $|W|$).

3.9 Simulation Results

3.9.1 Simulation Descriptions

In this section, we present results from simulations run in C++. In each case, we used $\mu = \tau = 1$ for the time and space discretization. We consider three different

scenarios: the first has no disturbance and no uncontrolled vehicles; the second has uncontrolled vehicles but no disturbance; the third has no uncontrolled vehicles but has a disturbance. We do not present a scenario which includes both uncontrolled vehicles and a disturbance, since these often result in empty solutions. Each scenario consists of an intersection with six vehicles, one on each of six roads arranged in a regular hexagonal pattern. Vehicles cross from one road to the road opposite their starting road. Specifically, if the set of vehicles is $\mathcal{N} = \{1, \ldots, 6\}$, then vehicle $i \in \mathcal{N}$ starts on road $r_{i,1} = i$ and ends on road $r_{i,2} = 1 + [(i + 2) \mod 6]$. Thus, the three pairs of vehicles $(1, 4)$, $(2, 5)$, and $(3, 6)$ can occupy the intersection simultaneously, but vehicles from different pairs cannot (see Fig. 3.6). Each of the three scenarios consisted of three "sub-scenarios". In each one, the length of the intersection was identical for all roads, but this length was changed for different sub-scenarios, with all other parameters remaining the same. For each sub-scenario, two simulations were run, one with the algorithm as described in Sec. 3.8, and one with this algorithm augmented with an optimization described in detail in the appendix. Briefly, this optimization consists of computing the capture set for each pair of vehicles that cannot simultaneously be inside the intersection. This can be done easily for such pairs of vehicles, since the bad set is bounded and convex in this case. We describe each scenario in more detail below.

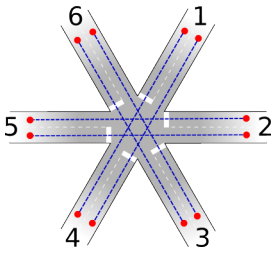


Figure 3.6: The intersection and vehicle paths used in each of the simulations of this section. Blue lines are drawn for each vehicle indicating starting road and ending road.

Scenario 1: In each of the three sub-scenarios, all vehicles' starting positions are at a distance of 26 from the end of the intersection and the vehicles have available velocities $\{1, 2\}$. The parameters for the first sub-scenario are such that the vehicles have just enough space to get through the intersection safely. Specifically, one pair of vehicles must travel at maximal velocity until they have crossed the intersection, a second pair of vehicles must enter the intersection as the first pair leaves and travel at maximal velocity while in the intersection, and the last pair of vehicles must travel at minimal velocity until they enter the intersection, which again occurs as the previous pair leaves the intersection. The second and third sub-scenarios have shorter intersection lengths, resulting in a strictly larger set of solutions over the same state space.

Scenario 2: In each of the three sub-scenarios, vehicles 1 and 4 are uncontrolled whereas the other four vehicles are controlled. The controlled and uncontrolled vehicles start at a distance of 26 and 15, respectively, from the end of the intersection and all vehicles have available velocities $\{1, 2\}$. The parameters for the first sub-scenario are again such that the vehicles have just enough space to get through the intersection safely. Because the uncontrolled vehicles start closer to the intersection, and have the same set of available controls, they must cross the intersection first in any safe solution. The "worst-case scenario" therefore occurs when the uncontrolled vehicles maintain minimal velocity throughout the simulation, in which case one pair of controlled vehicles must enter the intersection as the uncontrolled vehicles leave and must travel at maximal velocity while inside the intersection. In this case, the last pair of controlled vehicles must maintain minimal velocity until entering the intersection, which again occurs as the previous pair leaves the intersection. Once again, the second and third sub-scenarios have shorter intersection lengths, resulting in a strictly larger set of solutions over the same state space.

Scenario 3: This scenario uses a disturbance in the range $[-1, 1]$. In each of the three sub-scenarios, vehicles 1 and 4 start at a distance of 12 from the end of the intersection, vehicles 2 and 5 at a distance of 8, and vehicles 3 and 6 at a distance of 4. All vehicles have available velocities $\{2, 3, 4, 5\}$. The parameters for the first sub-scenario are again such that the vehicles have just enough space to get through the intersection safely. The safe solutions in this case consist of allowing vehicles 3 and 6 to cross the intersection first, followed by vehicles 2 and 5, and then finally vehicles 1 and 4. Because of the disturbance, no control action can ensure a velocity of less than 3 or more than 4. Thus, the "worst-case scenario" occurs when vehicles 3 and 6 travel at velocity 4 and vehicles 1 and 4 travel at velocity 3, leaving vehicles 2 and 5 to navigate in between. As was the case in scenarios 1 and 2, the second and third sub-scenarios have shorter intersection lengths, resulting in a strictly larger set of solutions over the same state space.

3.9.2 Results & Analysis

For each simulation, we provide the number of states examined by the algorithm, the subset of those states that were safe, the product of the number of examined states with $|U_c||\mathbf{Post}_{u_c}(q)|$, and the running time in seconds. For each sub-scenario, we give the ratio of unsafe examined states for the simulation with optimization compared to that without optimization. This data is presented in Tables 3.1, 3.2, and 3.3.

A comparison of the last two rows shows that the algorithm can compute maximally permissive solutions in approximately one minute for systems with over 200 million accessible transitions in the absence of a disturbance, and with over 50 billion accessible transitions in the presence of a disturbance. There are two reasons why the algorithm can deal with systems with larger numbers of accessible transitions in the presence of a disturbance. First, the fact that there is only one safety test for each

Table 3.1: Scenario 1: No uncontrolled vehicles, no disturbance.

Int. Range	$\alpha = 3$		$\alpha = 2$		$\alpha = 1$	
Optimization?	×	√	×	√	×	√
Examined States	877886	37901	1815045	438432	3187418	2030613
Safe States	754		146097		1752458	
(Ex. States)$\lvert U_c \rvert \lvert \mathbf{Post}_{u_c}(q) \rvert$	5.62×10^7	2.43×10^6	1.16×10^8	2.81×10^7	2.04×10^8	1.30×10^8
Time (s)	6.61	0.37	16.32	5.55	44.75	38.59
Unsafe Ratio	0.0424		0.175		0.194	

$u_c \in U_c$, rather than for each combination of $u_c \in U_c$, $u_{uc} \in U_{uc}$, and $w \in W$ significantly reduces the number of safety tests. In simulation 3, $|\mathbf{Post}_{u_c}(q)| = 3^6 = 729$. Second, although the asymptotic complexity has a term of $O(|\tilde{Q}||U_c||\mathbf{Post}_{u_c}(q)|)$, this term derives from verifying if q' is deadlocked, for each successor $q' \in \mathbf{Post}_{u_c}(q)$. Because u_c is unsafe from q if there exists any such deadlocked q', the algorithm ceases to examine further members of $\mathbf{Post}_{u_c}(q)$ as soon as it finds any deadlocked successor (line 16 of Algorithm 2). This also explains why the running time increases more quickly in going from sub-scenario 1 to sub-scenario 3 in the presence of a disturbance: as the proportion of examined states that are safe increases, a larger number of control actions become safe, and the average number of successor states that must be examined on line 14 of Algorithm 2 increases more rapidly when $|\mathbf{Post}_{u_c}(q)|$ is large.

The results of all sub-scenarios show that the majority of unsafe states examined through the depth-first search without the optimization are states which are unsafe because there exists no solution for a particular pair of vehicles (at least 79% in all sub-scenarios). Nonetheless, the optimization does incur some overhead, so that the computational savings are greatest when there is a large portion of the reachable state space that consists of unsafe states. Because of this overhead, the running time is actually greater with optimization than without in sub-scenario 3 of scenario 2.

Remark III.30. In scenario 3, the number of safe examined states differs when comparing simulations with and without the optimization. This is due to safe states that are reached through non-determinism but which become unreachable in the final

Table 3.2: Scenario 2: Uncontrolled vehicles, no disturbance.

Int. Range	$\alpha = 3$		$\alpha = 2$		$\alpha = 1$	
Optimization?	×	√	×	√	×	√
Examined	1044346	360067	1680685	891391	2130800	1515525
Safe	229581		735974		1420771	
(Ex. States)$\lvert U_c \rvert \lvert \mathbf{Post}_{u_c}(q) \rvert$	6.68×10^7	2.30×10^7	1.08×10^8	5.70×10^7	1.36×10^8	9.70×10^7
Time (s)	5.74	3.39	12.76	11.66	20.36	22.87
Unsafe Ratio	0.160		0.165		0.133	

Table 3.3: Scenario 3: Disturbance, no uncontrolled vehicles.

Int. Range	$\alpha = 3/2$		$\alpha = 1$		$\alpha = 1/2$	
Optimization?	×	√	×	√	×	√
Examined	3385	1630	9850	5329	22866	16745
Safe	1575	1575	5623	5030	15179	15133
(Ex. States)$\lvert U_c \rvert \lvert \mathbf{Post}_{u_c}(q) \rvert$	1.01×10^{10}	4.87×10^9	2.94×10^{10}	1.59×10^{10}	6.83×10^{10}	5.00×10^{10}
Time (s)	1.06	0.75	12.03	9.15	76.83	58.37
Unsafe Ratio	0.030		0.071		0.210	

supervised system. As an example, suppose that $q', q'' \in \mathbf{Post}_{u_c}(q)$ and that q' is found to be safe, whereas q'' is found to be unsafe. Then $u_c \notin S(q)$, and q' could become unreachable, after having been examined by the algorithm. In some cases, these states will not be examined by the algorithm using the capture set optimization, resulting in a smaller number of examined safe states.

3.10 Conclusion

We considered the problem of supervising a set of vehicles approaching an intersection so as to avoid collisions, in the presence of environmental uncertainty in the form of uncontrolled vehicles and a disturbance. We solved this problem by constructing a DES abstraction and leveraging supervisory control methods of DES, a natural formulation for problems involving uncontrolled elements in which it is desired to obtain maximally permissive safe and non-deadlocking supervisors. We described the state reduction and exact state reduction relations between systems and abstractions, and used these to show that translating the supervisor for the abstraction back to the original problem domain preserves not only safety and non-deadlockingness, but also maximal permissiveness. Finally, we presented a customized algorithm for solving this supervisory control problem, and demonstrated its scalability through simula-

tion. This works extends the range of applications of DES. Moreover, to the best of our knowledge, it is the first DES application where the discrete event model is obtained by building a state reduction abstraction of the underlying continuous system model. Future work includes the extension of this work to the case of measurement uncertainty, second order dynamics, and stochastic problem formulations.

Appendix : Equations for Checking Safety

This appendix provides the equations that were used in the simulations of Sec. 3.9 for verifying the safety of a DES transitions (Part 1), and the equations for the pairwise capture sets for vehicles that cannot simultaneously be inside the intersection (Part 2).

Part 1: Verifying if $A_{q,u_c}(t) \cap B = \emptyset$ for all $t \in [0, \tau]$.

In part 1 of this appendix, we prove the equations used for verifying the safety of transitions. As stated in Sec. 3.8, there are equations for each pair of vehicles $i, j \in \mathcal{N}$, and verifying the safety of a DES transition for some initial state $q \in \tilde{Q}$ and $u_c \in U_c$ is done by verifying the corresponding equations for each pair of vehicles. We consider three cases (see Sec. 3.4): $x_i, x_j \leq 0$, $|x_i - x_j| < \gamma$ (case 1a), $x_i, x_j \geq 0$, $|x_i - x_j| < \gamma$ (case 1b), and $[-\alpha_{r_{i,1}} < x_i < \alpha_{r_{i,2}}] \wedge [-\alpha_{r_{j,1}} < x_j < \alpha_{r_{j,2}}]$ (case 2). The equations for these cases are provided in Props. (III.33)-(III.35), respectively. Note that there is no "case 1c" when $x_i \leq 0$ and $x_j \geq 0$, since the vehicles would then be on different roads.

We begin by defining the set $A_{q,u_c}([0,\tau]) := \bigcup_{t\in[0,\tau]} A_{q,u_c}(t)$. Because the bad set is defined as a union of sets of linear inequalities, with one set for each pair of vehicles, we verify $A_{q,u_c}([0,\tau]) \cap B = \emptyset$ by considering each pair of vehicles in turn. For any vehicle $i \in \mathcal{N}$ and any set $P \subseteq X$, let $\pi_i(P)$ denote the projection of P onto the i axis. Similarly, for any pair of vehicles $i, j \in \mathcal{N}$ and a set $P \subseteq X$, let $\pi_{i,j}(P)$ denote the projection of P onto the $i - j$ plane.

Proposition III.31. $(x_i, x_j) \in \pi_{i,j}(A_{q,u_c}([0,\tau]))$ *iff all of the following inequalities hold:*

$$x_i > q_i - \mu\tau/2 \tag{3.21}$$

$$x_j > q_j - \mu\tau/2 \tag{3.22}$$

$$x_i \leq q_i + \mu\tau/2 + \overline{v}_{u_c,i}\tau \tag{3.23}$$

$$x_j \leq q_j + \mu\tau/2 + \overline{v}_{u_c,j}\tau \tag{3.24}$$

$$\overline{v}_{u_c,i}(x_j - q_j + \mu\tau/2) - \underline{v}_{u_c,j}(x_i - q_i - \mu\tau/2) > 0 \tag{3.25}$$

$$\overline{v}_{u_c,j}(x_i - q_i + \mu\tau/2) - \underline{v}_{u_c,i}(x_j - q_j - \mu\tau/2) > 0 \tag{3.26}$$

Proof. From Eqs. (3.17), (3.18) and the assumption that $v_{min} + d_{min} \geq \mu > 0$, we have that $\pi_i(A_{q,u_c}(t)) = (q_i - \mu\tau/2 + \underline{v}_{u_c,i}t, q_i + \mu\tau/2 + \overline{v}_{u_c,i}t]$ is an interval whose lower and upper bounds are increasing in time, for every $i \in \mathcal{N}$. It follows that the set $\{t \in \mathbb{R} : x_i \in \pi_i(A_{q,u_c}(t)\}$ will have the form $[t_{i,min}, t_{i,max})$, where $t_{i,min} := \inf\{t \in \mathbb{R} : x_i \in \pi_i(A_{q,u_c}(t))\}$ and $t_{i,max} := \sup\{t \in \mathbb{R} : x_i \in \pi_i(A_{q,u_c}(t))\}$ are given by:

$$t_{i,min} = \frac{x_i - q_i - \mu\tau/2}{\overline{v}_{u_c,i}} \tag{3.27}$$

$$t_{i,max} = \frac{x_i - q_i + \mu\tau/2}{\underline{v}_{u_c,i}} \tag{3.28}$$

Now define $t_{j,min}$ and $t_{j,max}$ analogously to $t_{i,min}$ and $t_{i,max}$. Then:

$$\begin{aligned}
& \exists t \in [0,\tau] \text{ s.t. } [x_i \in \pi_i(A_{q,u_c}(t))] \wedge [x_j \in \pi_j(A_{q,u_c}(t))] \\
\Leftrightarrow\ & [0,\tau] \cap [t_{i,min}, t_{i,max}) \cap [t_{j,min}, t_{j,max}) \neq \emptyset \\
\Leftrightarrow\ & [t_{i,max} > 0] \wedge [t_{j,max} > 0] \wedge [t_{i,min} \leq \tau] \wedge [t_{j,min} \leq \tau] \\
& \quad \wedge [t_{j,max} > t_{i,min}] \wedge [t_{i,max} > t_{j,min}]
\end{aligned}$$

and these last six inequalities give Eqs. (3.21)-(3.26), in order. □

As stated above, we can check if $A_{q,u_c}([0,\tau]) \cap B = \emptyset$ by considering each pair of vehicles in turn. There are three types of constraints to consider:

Case 1a: $x_i, x_j \leq 0$, $|x_i - x_j| < \gamma$.

Lemma III.32. *Consider any* $\underline{x}_i, \overline{x}_i, \underline{x}_j, \overline{x}_j \in \mathbb{R}$. *Then:*

$$\begin{aligned}&(\exists x_i \in (\underline{x}_i, \overline{x}_i])(\exists x_j \in (\underline{x}_j, \overline{x}_j])(x_i \leq 0 \wedge x_j \leq 0 \wedge |x_i - x_j| < \gamma) \\ &\Leftrightarrow [\underline{x}_i < \overline{x}_i \wedge \underline{x}_i < 0 \wedge \underline{x}_j < \overline{x}_j \wedge \underline{x}_j < 0 \wedge \underline{x}_i - \overline{x}_j < \gamma \wedge \underline{x}_j - \overline{x}_i < \gamma]\end{aligned} \tag{3.29}$$

Proof. ($\Rightarrow$):

$$\begin{aligned} x_i \in (\underline{x}_i, \overline{x}_i] \Rightarrow \underline{x}_i < x_i \leq \overline{x}_i \quad &\Rightarrow \quad \underline{x}_i < \overline{x}_i \\ \underline{x}_i < x_i \wedge x_i \leq 0 \quad &\Rightarrow \quad \underline{x}_i < 0 \\ x_j \in (\underline{x}_j, \overline{x}_j] \Rightarrow \underline{x}_j < x_j \leq \overline{x}_j \quad &\Rightarrow \quad \underline{x}_j < \overline{x}_j \\ \underline{x}_j < x_j \wedge x_j \leq 0 \quad &\Rightarrow \quad \underline{x}_j < 0 \\ x_i - x_j < \gamma \wedge \underline{x}_i < x_i \wedge x_j \leq \overline{x}_j \quad &\Rightarrow \quad \underline{x}_i - \overline{x}_j < \gamma \\ x_j - x_i < \gamma \wedge \underline{x}_j < x_j \wedge x_i \leq \overline{x}_i \quad &\Rightarrow \quad \underline{x}_j - \overline{x}_i < \gamma \end{aligned}$$

($\Leftarrow$) It cannot be that both $\underline{x}_i - \underline{x}_j \geq \gamma$ and $\underline{x}_j - \underline{x}_i \geq \gamma$, as this would imply $0 \geq 2\gamma > 0$. Thus, at least one of $\underline{x}_i - \underline{x}_j < \gamma$, or $\underline{x}_j - \underline{x}_i < \gamma$ holds. If they both hold, we may take $x_i = \underline{x}_i + \epsilon$ and $x_j = \underline{x}_j + \epsilon$ for some sufficiently small $\epsilon > 0$ and we are done. Suppose without loss of generality then that $\underline{x}_i - \underline{x}_j < \gamma$ but $\underline{x}_j - \underline{x}_i \geq \gamma$. Let $\overline{x}_i = \underline{x}_j - \gamma$. Thus, $\underline{x}_j - \overline{x}_i = \gamma$, $\overline{x}_i - \underline{x}_j = -\gamma < \gamma$ and $\overline{x}_i < 0$ (since $\underline{x}_j < 0$). We may therefore take $x_i = \overline{x}_i + \epsilon$ and $x_j = \underline{x}_j + \epsilon$ for some sufficiently small $\epsilon > 0$ and

we are done. □

Proposition III.33. *The set* $\{(x_i, x_j) \in \pi_{i,j}(A_{q,u_c}([0,\tau])) : x_i, x_j \leq 0 \wedge |x_i - x_j| < \gamma\}$ *is non-empty iff all of the following inequalities hold:*

$$q_i < \mu\tau/2 \tag{3.30}$$

$$q_j < \mu\tau/2 \tag{3.31}$$

$$\underline{v}_{u_c,j}(q_i + \mu\tau/2 + \gamma) - \max\{\overline{v}_{u_c,i}, \underline{v}_{u_c,j}\}(q_j - \mu\tau/2) > 0 \tag{3.32}$$

$$\underline{v}_{u_c,i}(q_j + \mu\tau/2 + \gamma) - \max\{\overline{v}_{u_c,j}, \underline{v}_{u_c,i}\}(q_i - \mu\tau/2) > 0 \tag{3.33}$$

$$[q_i + \mu\tau/2 + \gamma + \tau\max\{\overline{v}_{u_c,i}, \underline{v}_{u_c,j}\}] - [q_j - \mu\tau/2 + \tau\underline{v}_{u_c,j}] > 0 \tag{3.34}$$

$$[q_j + \mu\tau/2 + \gamma + \tau\max\{\overline{v}_{u_c,j}, \underline{v}_{u_c,i}\}] - [q_i - \mu\tau/2 + \tau\underline{v}_{u_c,i}] > 0 \tag{3.35}$$

Proof. Let $\pi_i(A_{q,u_c}(t)) = (\underline{x}_i(t), \overline{x}_i(t)]$ and $\pi_j(A_{q,u_c}(t)) = (\underline{x}_j(t), \overline{x}_j(t)]$. By Lemma III.32, it is necessary and sufficient to find some $t \in [0,\tau]$ such that $\underline{x}_i(t) < 0$, $\underline{x}_j(t) < 0$, $\underline{x}_i(t) - \overline{x}_j(t) < \gamma$, and $\underline{x}_j(t) - \overline{x}_i(t) < \gamma$. Now define $t_{i,max}$, $t_{j,max}$, t_{i-j}, and t_{j-i} by $\underline{x}_i(t_{i,max}) = 0$, $\underline{x}_j(t_{j,max}) = 0$, $\underline{x}_i(t_{i-j}) - \overline{x}_j(t_{i-j}) = \gamma$, and $\underline{x}_j(t_{j-i}) - \overline{x}_i(t_{j-i}) = \gamma$. These are given by:

$$t_{i,max} = -\frac{q_i - \mu\tau/2}{\underline{v}_{u_c,i}} \tag{3.36}$$

$$t_{j,max} = -\frac{q_j - \mu\tau/2}{\underline{v}_{u_c,j}} \tag{3.37}$$

$$t_{i-j} = \frac{(q_i - \mu\tau/2) - (q_j + \mu\tau/2 + \gamma)}{\overline{v}_{u_c,j} - \underline{v}_{u_c,i}} \tag{3.38}$$

$$t_{j-i} = \frac{(q_j - \mu\tau/2) - (q_i + \mu\tau/2 + \gamma)}{\overline{v}_{u_c,i} - \underline{v}_{u_c,j}} \tag{3.39}$$

Obviously, t_{i-j} is only well defined when $\overline{v}_{u_c,j} \neq \underline{v}_{u_c,i}$ and t_{j-i} is only well defined

when $\overline{v}_{u_c,i} \neq \underline{v}_{u_c,j}$. Because $\underline{x}_i(t)$ and $\underline{x}_j(t)$ are increasing in time, we have that:

$$\underline{x}_i(t) < 0 \quad \Leftrightarrow \quad t < t_{i,max} \tag{3.40}$$

$$\underline{x}_j(t) < 0 \quad \Leftrightarrow \quad t < t_{j,max} \tag{3.41}$$

On the other hand, $\underline{x}_i(t) - \overline{x}_j(t)$ is increasing in time if $\overline{v}_{u_c,j} < \underline{v}_{u_c,i}$, decreasing in time if $\overline{v}_{u_c,j} > \underline{v}_{u_c,i}$, and constant if $\overline{v}_{u_c,j} = \underline{v}_{u_c,i}$. It therefore follows that:

$$\underline{x}_i(t) - \overline{x}_j(t) < \gamma \Leftrightarrow \begin{cases} t < t_{i-j}, & \overline{v}_{u_c,j} < \underline{v}_{u_c,i} \\ t > t_{i-j}, & \overline{v}_{u_c,j} > \underline{v}_{u_c,i} \\ (q_j + \mu\tau/2 + \gamma) > (q_i - \mu\tau/2), & \overline{v}_{u_c,j} = \underline{v}_{u_c,i} \end{cases} \tag{3.42}$$

Similarly,

$$\underline{x}_j(t) - \overline{x}_i(t) < \gamma \Leftrightarrow \begin{cases} t < t_{j-i}, & \overline{v}_{u_c,i} < \underline{v}_{u_c,j} \\ t > t_{j-i}, & \overline{v}_{u_c,i} > \underline{v}_{u_c,j} \\ (q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2), & \overline{v}_{u_c,i} = \underline{v}_{u_c,j} \end{cases} \tag{3.43}$$

This would give nine cases to consider, but three are impossible, since $\overline{v}_{u_c,j} < \underline{v}_{u_c,i} \Rightarrow \underline{v}_{u_c,j} \leq \overline{v}_{u_c,j} < \underline{v}_{u_c,i} \leq \overline{v}_{u_c,i} \Rightarrow \underline{v}_{u_c,j} < \overline{v}_{u_c,i}$ and similarly, $\overline{v}_{u_c,i} < \underline{v}_{u_c,j} \Rightarrow \underline{v}_{u_c,i} < \overline{v}_{u_c,j}$. We will consider each of the six remaining cases in turn, but first prove the following claims:

$$t_{j-i} < t_{j,max} \wedge t_{i,max} > 0 \quad \Rightarrow \quad t_{j-i} < t_{i,max} \tag{3.44}$$

$$t_{i-j} < t_{i,max} \wedge t_{j,max} > 0 \quad \Rightarrow \quad t_{i-j} < t_{j,max} \tag{3.45}$$

$$t_{i-j} > 0 \wedge \overline{v}_{u_c,j} < \underline{v}_{u_c,i} \quad \Rightarrow \quad t_{j-i} < t_{i-j} \tag{3.46}$$

$$t_{j-i} > 0 \wedge \overline{v}_{u_c,i} < \underline{v}_{u_c,j} \quad \Rightarrow \quad t_{i-j} < t_{j-i} \tag{3.47}$$

Clearly, Eq. (3.44) holds if $t_{j-i} \leq 0$. If $t_{j-i} > 0$, then $\underline{x}_i(t_{j-i}) < \overline{x}_i(t_{j-i}) =$

$\underline{x}_j(t_{j-i}) - \gamma < \underline{x}_j(t_{j-i})$. From Eq. (3.41), we have that $t_{j-i} < t_{j,max} \Leftrightarrow \underline{x}_j(t_{j-i}) < 0$. Hence, $\underline{x}_i(t_{j-i}) < \underline{x}_j(t_{j-i}) < 0$ and therefore $t_{j-i} < t_{i,max}$ follows from Eq. (3.40), proving Eq. (3.44). Eq. (3.45) is proven similarly. To prove Eq. (3.46), suppose to the contrary that $t_{j-i} \geq t_{i-j} > 0$. As before, $t_{j-i} > 0 \Rightarrow \underline{x}_i(t_{j-i}) < \underline{x}_j(t_{j-i})$. From $\overline{v}_{u_c,j} < \underline{v}_{u_c,i}$, $t_{j-i} \geq t_{i-j}$, and Eq. (3.42), we have that $\underline{x}_i(t_{j-i}) \geq \overline{x}_j(t_{j-i}) + \gamma > \overline{x}_j(t_{j-i})$. Thus we have $\underline{x}_j(t_{j-i}) > \underline{x}_i(t_{j-i}) > \overline{x}_j(t_{j-i})$, which is a contradiction since it cannot be that $\underline{x}_j(t_{j-i}) > \overline{x}_j(t_{j-i})$ for $t_{j-i} > 0$, proving Eq. (3.46). Eq. (3.47) is proven similarly. We now proceed with the six cases. In what follows, note that Eqs. (3.32) and (3.34) both reduce to $(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)$ when $\overline{v}_{u_c,i} \leq \underline{v}_{u_c,j}$ and that Eqs. (3.33) and (3.35) similarly both reduce to $(q_j + \mu\tau/2 + \gamma) > (q_i - \mu\tau/2)$ when $\overline{v}_{u_c,j} \leq \underline{v}_{u_c,i}$.

Case (i): $\overline{v}_{u_c,j} = \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} = \underline{v}_{u_c,j}$.

$$
\begin{aligned}
& \exists t \in [0,\tau] \text{ s.t. } [\underline{x}_i(t) < 0] \wedge [\underline{x}_j(t) < 0] \\
& \quad \wedge [\underline{x}_i(t) - \overline{x}_j(t) < \gamma] \wedge [\underline{x}_j(t) - \overline{x}_i(t) < \gamma] \\
& \quad [0,\tau] \cap (-\infty, t_{i,max}) \cap (-\infty, t_{j,max}) \neq \emptyset \\
\Leftrightarrow \quad & \quad \wedge [(q_j + \mu\tau/2 + \gamma) > (q_i - \mu\tau/2)] \qquad \text{(Eqs. (3.40)-(3.43))} \\
& \quad \wedge [(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \\
& \quad [0 < t_{i,max}] \wedge [0 < t_{j,max}] \\
\Leftrightarrow \quad & \quad \wedge [(q_j + \mu\tau/2 + \gamma) > (q_i - \mu\tau/2)] \\
& \quad \wedge [(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \\
\Leftrightarrow \quad & [(3.30)] \wedge [(3.31)] \wedge [(3.33) \wedge (3.35)] \wedge [(3.32) \wedge (3.34)]
\end{aligned}
$$

Case (ii): $\overline{v}_{u_c,j} > \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} = \underline{v}_{u_c,j}$.

$$
\begin{aligned}
& \exists t \in [0,\tau] \text{ s.t. } [\underline{x}_i(t) < 0] \wedge [\underline{x}_j(t) < 0] \\
& \quad \wedge [\underline{x}_i(t) - \overline{x}_j(t) < \gamma] \wedge [\underline{x}_j(t) - \overline{x}_i(t) < \gamma] \\
\Leftrightarrow \quad & [0,\tau] \cap (-\infty, t_{i,max}) \cap (-\infty, t_{j,max}) \cap (t_{i-j}, \infty) \neq \emptyset \\
& \quad \wedge [(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \qquad \text{(Eqs. (3.40)-(3.43))} \\
\Leftrightarrow \quad & [0 < t_{i,max}] \wedge [0 < t_{j,max}] \wedge [t_{i-j} < \tau] \wedge [t_{i-j} < t_{i,max}] \\
& \quad \wedge [t_{i-j} < t_{j,max}] \wedge [(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \\
\Leftrightarrow \quad & [0 < t_{i,max}] \wedge [0 < t_{j,max}] \wedge [t_{i-j} < \tau] \wedge [t_{i-j} < t_{i,max}] \\
& \quad \wedge [(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \qquad \text{(Eq. (3.45))} \\
\Leftrightarrow \quad & [(3.30)] \wedge [(3.31)] \wedge [(3.35)] \wedge [(3.33)] \wedge [(3.32) \wedge (3.34)]
\end{aligned}
$$

Case (iii): $\overline{v}_{u_c,j} = \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} > \underline{v}_{u_c,j}$.

This is case is symmetrical to Case (ii).

Case (iv): $\overline{v}_{u_c,j} < \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} > \underline{v}_{u_c,j}$.

$$
\begin{aligned}
& \exists t \in [0,\tau] \text{ s.t. } [\underline{x}_i(t) < 0] \wedge [\underline{x}_j(t) < 0] \\
& \quad \wedge [\underline{x}_i(t) - \overline{x}_j(t) < \gamma] \wedge [\underline{x}_j(t) - \overline{x}_i(t) < \gamma] \\
\Leftrightarrow \quad & [0,\tau] \cap (-\infty, t_{i,max}) \cap (-\infty, t_{j,max}) \cap (-\infty, t_{i-j}) \cap (t_{j-i}, \infty) \neq \emptyset \qquad \text{(Eqs. (3.40)-(3.43))} \\
\Leftrightarrow \quad & [0 < t_{i,max}] \wedge [0 < t_{j,max}] \wedge [0 < t_{i-j}] \wedge [t_{j-i} < \tau] \\
& \quad \wedge [t_{j-i} < t_{i,max}] \wedge [t_{j-i} < t_{j,max}] \wedge [t_{j-i} < t_{i-j}] \\
\Leftrightarrow \quad & [0 < t_{i,max}] \wedge [0 < t_{j,max}] \wedge [0 < t_{i-j}] \wedge [t_{j-i} < \tau] \wedge [t_{j-i} < t_{j,max}] \qquad \text{(Eqs. (3.44), (3.46))} \\
\Leftrightarrow \quad & [(3.30)] \wedge [(3.31)] \wedge [(3.33) \wedge (3.35)] \wedge [(3.34)] \wedge [(3.32)]
\end{aligned}
$$

Case (v): $\overline{v}_{u_c,j} > \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} < \underline{v}_{u_c,j}$.

This is case is symmetrical to Case (iv).

Case (vi): $\overline{v}_{u_c,j} > \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} > \underline{v}_{u_c,j}$.

$$
\begin{aligned}
& \exists t \in [0,\tau] \text{ s.t. } [\underline{x}_i(t) < 0] \wedge [\underline{x}_j(t) < 0] \\
& \quad \wedge [\underline{x}_i(t) - \overline{x}_j(t) < \gamma] \wedge [\underline{x}_j(t) - \overline{x}_i(t) < \gamma] \\
\Leftrightarrow \quad & [0,\tau] \cap (-\infty, t_{i,max}) \cap (-\infty, t_{j,max}) \cap (t_{i-j}, \infty) \cap (t_{j-i}, \infty) \neq \emptyset \quad \text{(Eqs. (3.40)-(3.43))} \\
& [0 < t_{i,max}] \wedge [0 < t_{j,max}] \\
\Leftrightarrow \quad & \quad \wedge [t_{j-i} < \tau] \wedge [t_{j-i} < t_{i,max}] \wedge [t_{j-i} < t_{j,max}] \\
& \quad \wedge [t_{i-j} < \tau] \wedge [t_{i-j} < t_{i,max}] \wedge [t_{i-j} < t_{j,max}] \\
& [0 < t_{i,max}] \wedge [0 < t_{j,max}] \\
\Leftrightarrow \quad & \quad \wedge [t_{j-i} < \tau] \wedge [t_{j-i} < t_{j,max}] \qquad \text{(Eqs. (3.44), (3.45))} \\
& \quad \wedge [t_{i-j} < \tau] \wedge [t_{i-j} < t_{i,max}] \\
\Leftrightarrow \quad & [(3.30)] \wedge [(3.31)] \wedge [(3.34)] \wedge [(3.32)] \wedge [(3.35)] \wedge [(3.33)]
\end{aligned}
$$

□

Case 1b: $x_i, x_j \geq 0$, $|x_i - x_j| < \gamma$.

Proposition III.34. *The set* $\{(x_i, x_j) \in \pi_{i,j}(A_{q,u_c}([0,\tau])) : x_i, x_j \geq 0 \wedge |x_i - x_j| < \gamma\}$ *is non-empty iff all of the following inequalities hold:*

$$q_i \geq -\mu\tau/2 - \overline{v}_{u_c,i}\tau \tag{3.48}$$

$$q_j \geq -\mu\tau/2 - \overline{v}_{u_c,j}\tau \tag{3.49}$$

$$\begin{aligned} & \max\{\overline{v}_{u_c,i}, \underline{v}_{u_c,j}\}(q_i + \mu\tau/2 + \tau\overline{v}_{u_c,i}) \\ & \quad -\overline{v}_{u_c,i}(q_j - \mu\tau/2 - \gamma + \tau\underline{v}_{u_c,j}) > 0 \end{aligned} \tag{3.50}$$

$$\begin{aligned} & \max\{\overline{v}_{u_c,j}, \underline{v}_{u_c,i}\}(q_j + \mu\tau/2 + \tau\overline{v}_{u_c,j}) \\ & \quad -\overline{v}_{u_c,j}(q_i - \mu\tau/2 - \gamma + \tau\underline{v}_{u_c,i}) > 0 \end{aligned} \tag{3.51}$$

$$\begin{aligned} & (q_i + \mu\tau/2 + \tau\max\{\underline{v}_{u_c,j}, \overline{v}_{u_c,i}\}) \\ & \quad -(q_j - \mu\tau/2 - \gamma + \tau\underline{v}_{u_c,j}) > 0 \end{aligned} \tag{3.52}$$

$$\begin{aligned} & (q_j + \mu\tau/2 + \tau\max\{\underline{v}_{u_c,i}, \overline{v}_{u_c,j}\}) \\ & \quad -(q_i - \mu\tau/2 - \gamma + \tau\underline{v}_{u_c,i}) > 0 \end{aligned} \tag{3.53}$$

Proof. The proof is similar to that of Prop. III.33, and is omitted. □

Case 2: $[-\alpha_{r_{i,1}} < x_i < \alpha_{r_{i,2}}] \wedge [-\alpha_{r_{j,1}} < x_j < \alpha_{r_{j,2}}]$.

Proposition III.35. *The set* $\{(x_i, x_j) \in \pi_{i,j}(A_{q,u_c}([0,\tau])) : [-\alpha_{r_{i,1}} < x_i < \alpha_{r_{i,2}}] \wedge [-\alpha_{r_{j,1}} < x_j < \alpha_{r_{j,2}}]\}$ *is non-empty iff all of the following inequalities hold:*

$$q_i < \alpha_{r_{i,2}} + \mu\tau/2 \tag{3.54}$$

$$q_j < \alpha_{r_{j,2}} + \mu\tau/2 \tag{3.55}$$

$$q_i > -\alpha_{r_{i,1}} - \mu\tau/2 - \overline{v}_{u_c,i}\tau \tag{3.56}$$

$$q_j > -\alpha_{r_{j,1}} - \mu\tau/2 - \overline{v}_{u_c,j}\tau \tag{3.57}$$

$$\underline{v}_{u_c,j}(q_i + \mu\tau/2 + \alpha_{r_{i,1}}) - \overline{v}_{u_c,i}(q_j - \mu\tau/2 - \alpha_{r_{j,2}}) > 0 \tag{3.58}$$

$$\underline{v}_{u_c,i}(q_j + \mu\tau/2 + \alpha_{r_{j,1}}) - \overline{v}_{u_c,j}(q_i - \mu\tau/2 - \alpha_{r_{i,2}}) > 0 \tag{3.59}$$

Proof. We proceed similarly to the proof of Prop. III.31. From Eqs. (3.17), (3.18) and the assumption that $v_{min} + d_{min} \geq \mu > 0$, we have that $\pi_i(A_{q,u_c}(t)) = (q_i - \mu\tau/2 + \underline{v}_{u_c,i}t, q_i + \mu\tau/2 + \overline{v}_{u_c,i}t]$ is an interval whose lower and upper bounds are increasing in time, for every $i \in \mathcal{N}$. It follows that the set $\{t \in \mathbb{R} : (-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \cap \pi_i(A_{q,u_c}(t)) \neq \emptyset\}$ will have the form $(t^2_{i,min}, t^2_{i,max})$, where $t^2_{i,min} := \inf\{t \in \mathbb{R} : (-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \cap \pi_i(A_{q,u_c}(t)) \neq \emptyset\}$ and $t^2_{i,max} := \sup\{t \in \mathbb{R} : (-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \cap \pi_i(A_{q,u_c}(t)) \neq \emptyset\}$ are given by:

$$t^2_{i,min} = \frac{-q_i - \alpha_{r_{i,1}} - \mu\tau/2}{\overline{v}_{u_c,i}} \tag{3.60}$$

$$t^2_{i,max} = \frac{-q_i + \alpha_{r_{i,2}} + \mu\tau/2}{\underline{v}_{u_c,i}} \tag{3.61}$$

Now define $t^2_{j,min}$ and $t^2_{j,max}$ analogously to $t^2_{i,min}$ and $t^2_{i,max}$. Then:

$$\begin{aligned}
& \exists t \in [0,\tau] \text{ s.t. } [(-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \cap \pi_i(A_{q,u_c}(t))] \wedge [(-\alpha_{r_{j,1}}, \alpha_{r_{j,2}}) \cap \pi_j(A_{q,u_c}(t))] \\
\Leftrightarrow \quad & [0,\tau] \cap (t^2_{i,min}, t^2_{i,max}) \cap (t^2_{j,min}, t^2_{j,max}) \neq \emptyset \\
\Leftrightarrow \quad & [t^2_{i,max} > 0] \wedge [t^2_{j,max} > 0] \wedge [t^2_{i,min} < \tau] \wedge [t^2_{j,min} < \tau] \\
& \wedge t^2_{j,max} > t^2_{i,min} \wedge t^2_{i,max} > t^2_{j,min}
\end{aligned}$$

and these last six inequalities give Eqs. (3.54)-(3.59), in order. □

Part 2: The Capture Set Optimization

Here we describe an optimization which allows for a substantial reduction of the number of unsafe examined states in Alg. 2. The optimization is based on the observation that the bad set is convex (rectangular) for a pair of vehicles which cannot simultaneously be inside the intersection (Case 2 of Part 1). Thus it is straight-forward to compute the *capture set* of states from which no supervisor can ensure avoidance of the bad set for such a pair of vehicles. Before stating the theorem, we define the minimal and maximal velocities which can be forced by the supervisor, given that it does not control the uncontrolled vehicles or the disturbance:

$$\underline{v}^c_i = \begin{cases} v_{min} + d_{max}, & \text{vehicle } i \text{ is controlled} \\ v_{max} + d_{max}, & \text{vehicle } i \text{ is uncontrolled} \end{cases} \tag{3.62}$$

$$\overline{v}^c_i = \begin{cases} v_{max} + d_{min}, & \text{vehicle } i \text{ is controlled} \\ v_{min} + d_{min}, & \text{vehicle } i \text{ is uncontrolled} \end{cases} \tag{3.63}$$

Proposition III.36. *Given two vehicles i and j on different roads, there does not exist any safe and non-deadlocking supervisor $\sigma : \tilde{Q} \to 2^{U_c}$ with $\sigma(q) \neq \emptyset$, for any*

$q \in \tilde{Q}$ such that $\exists x \in \ell^{-1}(q)$ satisfying all of the following equations:

$$x_i < \alpha_{r_{i,2}} \tag{3.64}$$

$$x_j < \alpha_{r_{j,2}} \tag{3.65}$$

$$\overline{v}_i^c(x_j + \alpha_{r_{j,1}}) - \underline{v}_j^c(x_i - \alpha_{r_{i,2}}) > 0 \tag{3.66}$$

$$\overline{v}_j^c(x_i + \alpha_{r_{i,1}}) - \underline{v}_i^c(x_j - \alpha_{r_{j,2}}) > 0 \tag{3.67}$$

Proof. First, it follows from the definitions of $\underline{v}_i^c$ and $\overline{v}_i^c$ that, for any x satisfying Eqs. (3.64)-(3.67) and $u_c \in U_c$, there exists some $u_{uc} \in U_{uc}$ and $d : [0, \tau] \to D$ such that $x(t) = x + u(t/\tau) + d(t)$ either remains inside the set given by Eqs. (3.64)-(3.67) for $t \in [0, \tau]$, or enters the bad set for some $t \in [0, \tau]$ (see Fig. 3.7). Second, it follows from $v_{min} + d_{min} > 0$ that no control strategy can prevent the vehicles from eventually leaving the set given by Eqs (3.64)-(3.67). Thus either the system eventually reaches some state $q' \in \tilde{Q}$ such that $\sigma(q') = \emptyset$, or σ allows the system to enter the bad set. □

We can obtain the set of set of states q for which there exists some $x \in \ell^{-1}(q)$ satisfying Eqs. (3.64)-(3.67) by taking this set and "inflating it" by $\mu\tau/2$, to capture the effect of the discretization. This gives one of two possibilities, depending on whether the set of Eqs. (3.64)-(3.67) is open or closed. Then set will be open if $\frac{\overline{v}_j^c}{\underline{v}_i^c} \leq \frac{\underline{v}_j^c}{\overline{v}_i^c}$ and closed if $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$. If the set is open, the equations become:

$$q_i < \alpha_{r_{i,2}} + \mu\tau/2 \tag{3.68}$$

$$q_j < \alpha_{r_{j,2}} + \mu\tau/2 \tag{3.69}$$

$$\overline{v}_i^c(q_j + \alpha_{r_{j,1}} + \mu\tau/2) - \underline{v}_j^c(q_j - \alpha_{r_{i,2}} - \mu\tau/2) > 0 \tag{3.70}$$

$$\overline{v}_j^c(q_i + \alpha_{r_{i,1}} + \mu\tau/2) - \underline{v}_i^c(q_i - \alpha_{r_{j,2}} - \mu\tau/2) > 0 \tag{3.71}$$

If the set is closed, then two more equations must be added in general (see Fig. 3.7)

$$q_i > \frac{\overline{v}_i^c \overline{v}_j^c \alpha_{r_{i,1}} + \overline{v}_i^c \underline{v}_i^c \alpha_{r_{j,2}} + \overline{v}_i^c \underline{v}_i^c \alpha_{r_{j,1}} + \underline{v}_i^c \underline{v}_j^c \alpha_{r_{i,2}}}{\overline{v}_i^c \overline{v}_j^c - \underline{v}_i^c \underline{v}_j^c} - \mu\tau/2 \tag{3.72}$$

$$q_j > \frac{\overline{v}_i^c \overline{v}_j^c \alpha_{r_{j,1}} + \overline{v}_j^c \underline{v}_j^c \alpha_{r_{i,2}} + \overline{v}_j^c \underline{v}_j^c \alpha_{r_{i,1}} + \underline{v}_i^c \underline{v}_j^c \alpha_{r_{j,2}}}{\overline{v}_i^c \overline{v}_j^c - \underline{v}_i^c \underline{v}_j^c} - \mu\tau/2 \tag{3.73}$$

If d_{min} and d_{max} are integer multiples of μ, then it can be shown these last two equations become unnecessary. We first prove a lemma.

Lemma III.37. *If d_{min} and d_{max} are integer multiples of μ, $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$, and $q \in \tilde{Q}$ satisfies Eqs. (3.70) and (3.71) then, for any $u_c \in U_c$, there exists $q' \in \mathbf{Post}_{u_c}(q)$ that also satisfies Eqs. (3.70) and (3.71).*

Proof. First note from Eqs. (3.62) and (3.63) that, if either vehicle is uncontrolled, then $\frac{\overline{v}_j^c}{\underline{v}_i^c} \leq 1$ and $\frac{\underline{v}_j^c}{\overline{v}_i^c} \geq 1$, violating $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$. It follows that both vehicles are controlled, and that $\overline{v}_i^c = \overline{v}_j^c > \underline{v}_i^c = \underline{v}_j^c$. We prove the following claim:

Claim: For any $u_c \in U_c$, there exists some $d_i \in [d_{min}, d_{max}]$ such that $u_{c,i}/\tau + d_i \in [\underline{v}_i^c, \overline{v}_i^c]$ and $u_{c,i}/\tau + d_i$ is an integer multiple of μ.

It suffices to prove that, for any $u_c \in U_c$, $[\underline{v}_i^c - u_{c,i}/\tau, \overline{v}_i^c - u_{c,i}/\tau] \cap [d_{min}, d_{max}]$ contains some integral multiple of μ, since we may then take such a value as d_i. Clearly, $u_{c,i}/\tau \in [v_{min}, v_{max}]$, from which it follows that $\underline{v}_i^c - u_{c,i}/\tau = v_{min} + d_{max} - u_{c,i}/\tau \leq d_{max}$ and that $\overline{v}_i^c - u_{c,i}/\tau = v_{max} + d_{min} - u_{c,i}/\tau \geq d_{min}$. Thus, $[\underline{v}_i^c - u_{c,i}/\tau, \overline{v}_i^c - u_{c,i}/\tau] \cap [d_{min}, d_{max}]$ is non-empty. Since it is non-empty, there must be at least one of d_{min} and $\underline{v}_i^c - u_{c,i}/\tau$ in the intersection of the two sets. Since both d_{min} and $\underline{v}_i^c - u_{c,i}/\tau$ are multiples of μ, the claim is proven.

Constructing d_i and d_j as in the claim, we obtain

$$\frac{\overline{v}_j^c}{\underline{v}_i^c} \geq \frac{u_{c,j}/\tau + d_j}{u_{c,i}/\tau + d_i} \geq \frac{\underline{v}_j^c}{\overline{v}_i^c}.$$

It follows that we can take $w \in W$ such that $w_i = d_i\tau$ and $w_j = d_j\tau$, obtaining q'

with $q_i' = q_i + u_{c,i} + w_i$, $q_j' = q_j + u_{c,j} + w_j$ such that $q' \in \mathbf{Post}_{u_c}(q)$ satisfies Eqs. (3.70) and (3.71). □

Corollary III.38. *If d_{min} and d_{max} are integer multiples of μ then, given two vehicles i and j on different roads, there does not exist any safe and non-deadlocking supervisor $\sigma : \tilde{Q} \to 2^{U_c}$ with $\sigma(q) \neq \emptyset$, for any $q \in \tilde{Q}$ satisfying Eqs. (3.68)-(3.71) only (i.e., without satisfying Eqs. (3.72) and (3.73)), even when $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$.*

Proof. We have already shown that the result holds if $\frac{\overline{v}_j^c}{\underline{v}_i^c} \leq \frac{\underline{v}_j^c}{\overline{v}_i^c}$, or $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$ and q satisfies Eqs. (3.68)-(3.73). It remains to be shown that the result also holds if d_{min} and d_{max} are integer multiples of μ, $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$, and q satisfies Eqs. (3.68)-(3.71), but not Eqs. (3.72) and (3.73). Consider any $u_c \in U_c$. By Lemma III.37, there exists $q' \in \mathbf{Post}_{u_c}(q)$ that also satisfies Eqs. (3.70) and (3.71). There are now three cases to consider:

Case 1: q' satisfies Eqs. (3.68)-(3.73).

We have shown in this case there exists no safe and non-deadlocking supervisor from q'.

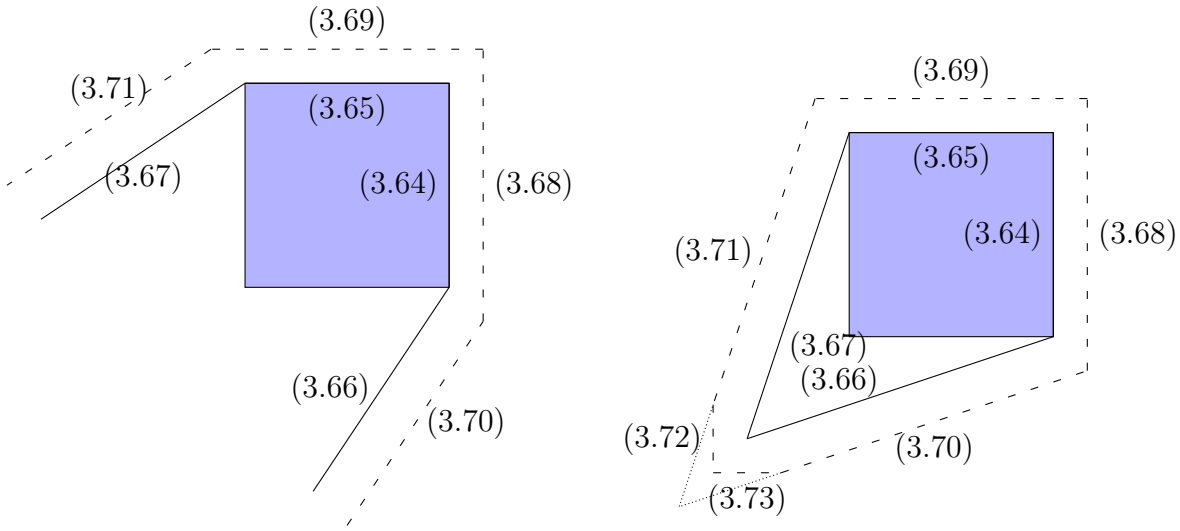
Case 2: q' satisfies Eqs. (3.68)-(3.71), but not both of Eqs. (3.72) and (3.73).

Because $d_{min} + v_{min} > 0$, Lemma III.37 can be applied repeatedly, until a q' is obtained which satisfies Eqs. (3.72) and (3.73).

Case 3: q' does not satisfy both of Eqs. (3.68) and (3.69).

In this case, the line segment from q to q' either crosses the bad set, or comes within a distance of $\mu\tau/2$ of it (see Fig. 3.7). In the latter case, we can find some pair $x \in \ell^{-1}(q)$ and $x' \in \ell^{-1}(q')$ such that the line segment from x to x' crosses the bad set. □

Figure 3.7 depicts the set described by Eqs. (3.64)-(3.67) of Prop. III.36, the inflated set of Eqs. (3.68)-(3.73), and the special case of Cor. III.38. The simulations of Sec. 3.9 satisfied the property that $d_{\min}$ and $d_{\max}$ were integer multiples of μ, and



(a) Open Case

(b) Closed Case

Figure 3.7: The capture sets of Eqs. (3.64)-(3.73) in the open (left) and closed (right) cases. The blue square denotes the bad set. The set of Eqs. (3.64)-(3.67) is depicted with solid lines, and its inflation by $\mu\tau/2$ is depicted in dashed lines. Right: If d_{min} and d_{max} are integer multiples of μ, then Eqs. (3.72) and (3.73) are unnecessary, which is shown by the dotted lines.

hence the code used Eqs. (3.68)-(3.71) only.

CHAPTER IV

Vehicle Control : The case of imperfect measurement

4.1 Abstract

We consider the problem of supervising a set of vehicles through an intersection, in the presence of uncontrolled vehicles, bounded disturbances, and measurement uncertainty. In real time, an estimate of the system's state is updated through a prediction-correction scheme, based on the actions of the controlled vehicles and the obtained imperfect measurements. The corrected estimates are mapped to a set of discrete states and sent to a supervisor, which outputs a set of allowable velocities for the controlled vehicles. The supervisor must be safe (i.e., collision-free), non-deadlocking (i.e., the vehicles must eventually cross the intersection), and maximally permissive with respect to the chosen discretization. We show how to construct a suitable Discrete Event System (DES) model of the prediction-correction estimator of the continuous system and associated specifications such that solving for the maximally permissive supervisor of the DES yields a supervisor for the original system that is safe, non-deadlocking, and maximally permissive. Building on previous work, we present two new types of system abstractions: the state estimate reduction and the exact state estimate reduction. We show that, when the DES model is a state

estimate reduction of the prediction-correction estimator, the supervisor constructed through the above procedure will be maximally permissive among memoryless supervisors. In the case of an exact state estimate reduction, the obtained supervisor will be maximally permissive among all supervisors, not merely memoryless ones.

4.2 Introduction

We consider the problem of supervising a set of vehicles through an intersection, in the presence of uncontrolled vehicles, bounded disturbances, and measurement uncertainty. Rather than choosing a particular control action from every state, the supervisor outputs a set of allowable control actions for the vehicles. The goal of this work is to construct such a supervisor that is safe (i.e., the vehicles do not collide), non-deadlocking (i.e., the vehicles eventually cross the intersection and do not reach states where the set of allowable actions by the supervisor is empty), and maximally permissive.

Approaches to the problem of controlling vehicles at an intersection generally fall into one of three categories: computation of maximally controlled invariant sets; scheduling techniques; and abstraction. Methods in the first category naturally result in supervisors that are safe, non-deadlocking, and maximally permissive, since a control action can be allowed if and only if it keeps the system inside the maximally controlled invariant set. Work in this category includes: *Hafner and Del Vecchio* (2011); *Verma and Del Vecchio* (2011). These methods can also handle disturbances, uncontrollability, and measurement uncertainty, but typically make assumptions such as convexity or order preserving dynamics, without which they do not scale well to systems with multiple dimensions.

Scheduling techniques work by treating the intersection as an indivisible resource to be allocated to different vehicles at different times. The general version of the decision problem is NP-hard, but takes polynomial time when each job (vehicle) requires

the resource (the intersection) for the same amount of time. Mapping the vehicle control problem to the polynomial time scheduling problem therefore amounts to either: assuming symmetries in the problem; or a problem relaxation when such symmetries are not satisfied. Work in this category includes: *Colombo and Del Vecchio* (2012), its extension to the case of dynamics with disturbances, *Bruni et al.* (2013), and its extension to the case of uncontrolled vehicles, *Ahn et al.* (2014). To our knowledge, these methods have not been extended to the case of measurement uncertainty. Furthermore, the assumption of mutual exclusiveness of the intersection's use is restrictive, as it precludes vehicles on common or non-intersecting trajectories (e.g., in the case of right turns) from utilizing the intersection simultaneously.

Our approach falls in the category of abstraction. This approach generally consists of the following three steps: mapping the original, large (usually infinite), state space of a problem instance to a smaller, finite, set of states and defining the transitions over the smaller space to model the possible behaviors of the original system; computing a controller or supervisor for the finite model; and refining this controller or supervisor to one for the original system. The type of relation ((bi)simulation, alternating (bi)simulation, and various approximate variations of these) between the original and abstracted system determines what kind of guarantees can be made on the refined controller or supervisor. Work in this category includes: *Colombo and Del Vecchio* (2011a,b); *Colombo and Girard* (2013), which use differential flatness to guarantee bounds on the distance between safe trajectories and trajectories allowed by the obtained supervisor. These methods have not been extended to the case of measurement uncertainty, as the presence of measurement uncertainty typically result in a loss of the aforementioned guaranteed bounds. Our work is most similar to that of *Girard et al.* (2010); *Pola and Tabuada* (2009); *Zamani et al.* (2012); *Camara et al.* (2011), which construct abstract models satisfying simulation or alternating simulation relations with the original system models. This work defines new types

of relations between systems and their abstractions, based on alternating simulation relations.

In this work, we show how to construct suitable DES abstractions of systems with safety and non-deadlocking specifications under the presence of measurement uncertainty. We then obtain a maximally permissive safe and non-deadlocking supervisor for the abstracted system by solving the basic supervisory control problem in the non-blocking case (BSCP-NB), *Ramadge and Wonham* (1987); *Cassandras and Lafortune* (2008). This supervisor, when applied to the continuous system, preserves safety and non-deadlockingness, as well as maximal permissiveness, with respect to the discretization of the abstraction. Given any control system with imperfect measurements, one can define a prediction-correction estimator for the system. Given an initial state estimate and a control action, this estimator *predicts* the possible sets of states after some duration τ, and then *corrects* this estimate upon a new measurement. In this work, we define two types of relations between these estimators and their abstractions: the state estimate reduction and the exact state estimate reduction. We show that, when an abstraction is a state estimate reduction of the estimator of the original system, the supervisor obtained through the procedure described at the beginning of this paragraph will be maximally permissive among memoryless supervisors. On the other hand, when the abstraction is an exact state estimate reduction of the estimator of the original system, the obtained supervisor will be maximally permissive among all supervisors, not merely memoryless ones.

This work extends our previous work in *Dallal et al.* (2014) in which we considered systems with perfect measurement. In that work, we defined the state reduction and exact state reduction relations between systems and abstractions and showed how to construct a DES abstraction G that is a state reduction of the system under consideration in the vehicle control problem (but in the case of perfect measurement), by discretizing the system in time and in space using a lattice. We also proved conditions

under which G would be an exact state reduction of the vehicle control system. The system G used controllable events to model the actions of the controlled vehicles and uncontrollable events to model the uncontrolled vehicles and the disturbance. In this work, we define a set of equivalence classes for the set of measurements, given the chosen discretization of the abstraction. These equivalence classes then constitute a new set of "measurement events" which are observable but uncontrollable, whereas the events modelling the uncontrolled vehicles and the disturbance are defined as unobservable and uncontrollable. This yields a new DES abstraction G' containing a mix of uncontrollable and unobservable events. We show in this work that the discretization parameters can be chosen so that the observer $\overline{G}$ of G' is a state estimate reduction of the prediction-correction estimator of the imperfectly measured vehicle control system. Furthermore, we show that when G is an exact state reduction of the perfectly measured vehicle control systems, then $\overline{G}$ will be an exact state estimate reduction of the prediction-correction estimator of the imperfectly measured vehicle control system. It is also important note that, for this system, state estimates will always be boxes (products of intervals), so that the determinization step in the construction of the observer only increases the state space quadratically, rather than exponentially.

The contributions of this work are, first, in the leveraging of supervisory control theory of DES, which is particularly well suited to finding maximally permissive solutions to control problems over finite state spaces subject to safety and non-blocking specifications. Second, the definitions of state estimate reduction and exact state estimate reduction are general concepts which allow for the construction of maximally permissive memoryless supervisors in the presence of measurement uncertainty. Preliminary versions of some of the results presented here have appeared in *Dallal et al.* (2013a), *Dallal et al.* (2013b).

The organization of this work is as follows. In section 4.3, we define the vehicle

control system and present the supervisory control problem to be solved. In section 4.4, we recall results from our previous work in *Dallal et al.* (2014), including the DES abstraction G of the perfectly measured case, the definitions of state reduction and exact state reduction, and the associated theorems. In section 4.5, we define the modified abstraction G' for the system with measurement uncertainty and proceed to define the notions of partially observed systems and estimator systems. Using these notions, we define the state estimate reduction and the exact state estimate reduction and prove associated theorems. In section 4.6, we prove results relating the (in)exact state reduction of the perfectly measured case to the (in)exact state estimate reduction of the imperfectly measured case. Finally, we conclude in section 4.7.

4.3 Model

Consider a set of vehicles $\mathcal{N} = \{1, \ldots, n\}$ with dynamics

$$\dot{x} = v + d, \tag{4.1}$$

where $x \in X$ is the vehicles' position, $v \in V$ is the control input, and $d \in D$ is a disturbance input. It is assumed that X is compact, that $D = [d_{min}, d_{max}]^n$ for some $d_{min}, d_{max} \in \mathbb{R}$ satisfying $d_{min} \leq 0 \leq d_{max}$, and that V is the set of vectors whose elements are in the set $\{a\mu, (a+1)\mu, \ldots, b\mu\}$, for some $a, b \in \mathbb{N}$. Denote values $a\mu$ and $b\mu$ by v_{min} and v_{max}, respectively. Let the set of vehicles be partitioned as $\mathcal{N} = \mathcal{N}_c \cup \mathcal{N}_{uc}$, where $\mathcal{N}_c$ is the set of controlled vehicles and $\mathcal{N}_{uc}$ is the set of uncontrolled vehicles (an uncontrolled vehicle is one whose actions cannot be constrained by the supervisor). With this partition, we write $V = V_c \times V_{uc}$ and $v = (v_c, v_{uc})$, for any $v \in V$.

We begin by discretizing system (4.1) in time with parameter $\tau \in \mathbb{R}_+$. Thus,

control actions are chosen at times $0, \tau, 2\tau, \ldots$ and kept constant for the following interval of duration τ. This gives the discrete-time system:

$$x_{k+1} = x_k + u_k + \delta_k, \tag{4.2}$$

where $x_k = x(k\tau)$, $u_k = v(k\tau)\tau$, and $\delta_k = \int_{k\tau}^{(k+1)\tau} d(t)dt$. If we define $U := V\tau$ and $\Delta := D\tau$, then we obtain $u \in U$ and $\delta \in \Delta$. As we did with V, we write $U = U_c \times U_{uc}$ and $u = (u_c, u_{uc})$, where $u_c \in U_c$ denotes the discrete action of the controlled vehicles and $u_{uc} \in U_{uc}$ denotes the discrete action of the uncontrolled vehicles.

We proceed to discretize the system in space by defining a set $\tilde{Q}$ of discrete states and a mapping $\ell : X \to \tilde{Q}$ as follows:

$$\ell_i(x_i) := \begin{cases} c\tau\mu, \text{ for } c \in \mathbb{Z} \text{ s.t.} & \\ c\tau\mu - \tau\mu/2 < x_i \leq c\tau\mu + \tau\mu/2, & \text{if } x_i \leq \alpha_{r_{i,2}} \\ q_{i,m}, & \text{if } x_i > \alpha_{r_{i,2}} \end{cases} \tag{4.3}$$

where $r_{i,2}$ is the road on which vehicle i leaves the intersection, $\alpha_{r_{i,2}}$ is the location of the end of the intersection on that road, and $q_{i,m}$ is a "special" state denoting that a vehicle has crossed the intersection. The function $\ell(x)$ is then defined as $\ell(x) := (\ell_1(x_1), \ldots, \ell_n(x_n))$. In words, $\ell(\cdot)$ maps continuous states to a lattice of discrete states with a spacing of $\tau\mu$ for vehicles before the end of the intersection. We denote by $q_m = (q_{1,m}, \ldots, q_{n,m})$ the unique discrete state where all vehicles have crossed the intersection. Assume that, for all $q \in \tilde{Q}$, there exists some $x \in X$ such that $\ell(x) = q$. Additionally, we define ℓ^{-1} and extend both ℓ and ℓ^{-1} to sets as follows: for any $q \in \tilde{Q}$, $\ell^{-1}(q) = \{x \in X : \ell(x) = q\}$; for any $I \subseteq X$, $\ell(I) = \bigcup_{x \in X} \ell(x)$; and for any $\iota \subseteq \tilde{Q}$, $\ell^{-1}(\iota) = \bigcup_{q \in \iota} \ell^{-1}(q)$.

We assume that the inputs of the uncontrolled vehicles and the disturbance are not directly observable. Instead, we obtain information about these through a state

measurement $\chi \in X$ satisfying $x \in L(\chi)$, where $L : X \to 2^X$ defines the set of states consistent with measurement χ. Given some maximal error $e_{max} \in \mathbb{R}_+$, L is defined by:

$$L(\chi) = [\chi - \mathbf{1}e_{max}, \chi + \mathbf{1}e_{max}], \tag{4.4}$$

where $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^n$ and, for any $a, b \in \mathbb{R}^n$, $[a, b] := \{x \in \mathbb{R}^n : a_i \leq x_i \leq b_i,\ i = 1, \ldots, n\}$ denotes a box in $\mathbb{R}^n$. With these definitions we can define a prediction-correction estimator for the discrete-time system of Eq. (4.2), consisting of the two functions $I^p : 2^X \times V_c \to 2^X$ and $I^c : 2^X \times X \to 2^X$ defined by:

$$I^p(I, v_c) = \bigcup_{x \in I, v_{uc} \in V_{uc}, \delta \in \Delta} (x + v\tau + \delta) \tag{4.5}$$

$$I^c(I, \chi) = I \cap L(\chi) \tag{4.6}$$

This prediction-correction estimator correctly computes the smallest state estimate compatible with the sequence of control inputs and measurements seen thus far (see e.g., *LaValle* (2006)). Additionally, we define L^{-1} and extend both L and L^{-1} to sets as follows: for any $x \in X$, $L^{-1}(x) = \{\chi \in X : x \in L(\chi)\}$; for any $I \subseteq X$, $L(I) = \bigcup_{\chi \in I} L(\chi)$; and for any $I \subseteq X$, $L^{-1}(I) = \bigcup_{x \in I} L^{-1}(x)$.

Finally, assume that there is some set B of bad states (representing collision points) and that we would like to define a supervisor so that $x(t) \notin B$, $\forall\, t \geq 0$. Let the set of roads be $\mathcal{R} = \{1, \ldots, m\}$. We assume that vehicle i travels on road $r_{i,1} \in \mathcal{R}$ for $x_i \leq 0$, and on road $r_{i,2} \in \mathcal{R}$ for $x_i \geq 0$ (i.e., vehicle i turns from road $r_{i,1}$ to road $r_{i,2}$ when $x_i = 0$). For road r, we take the intersection region to be of size α_r, so that vehicle i entering the intersection on road r is inside the intersection when $x_i \in (-\alpha_r, 0]$ and vehicle j exiting the intersection on road r is inside the intersection when $x_j \in [0, \alpha_r)$. We assume that the bad set has the form $B = \bigcup_{i,j \in \mathcal{N}, i \leq j} B_{ij}$, where B_{ij} has one of three forms:

- $B_{ij} = \emptyset$, if vehicles i and j are on non-intersecting paths.

- $B_{ij} = \{x \in X : x_i, x_j \leq 0 \wedge |x_i - x_j| < \gamma\}$ or $B_{ij} = \{x \in X : x_i, x_j \geq 0 \wedge |x_i - x_j| < \gamma\}$, if vehicles i and j enter the intersection on the same road or exit the intersection on the same road, respectively. Here $\gamma \in \mathbb{R}$ is a parameter defining the minimal separation distance between vehicles while on the same road.

- $B_{ij} = \{x \in X : x_i \in (-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \wedge x_j \in (-\alpha_{r_{j,1}}, \alpha_{r_{j,2}})\}$, if vehicles i and j's paths could result in a collision as the vehicles turn.

We wish to solve the following problem:

Problem IV.1. Let $2^X/\ell$ denote the quotient set of 2^X with respect to the equivalence relation $E \subseteq 2^X \times 2^X$ defined by $(I_1, I_2) \in E \Leftrightarrow \ell(I_1) = \ell(I_2)$. Given $\tilde{Q}$, define a supervisor $\sigma : 2^X/\ell \to 2^{V_c}$ that associates to each $I(k\tau) \subseteq X$ a set of inputs $v_c \in V_c$ allowed for the interval $[k\tau, (k+1)\tau]$ and constant over this time interval, with the following properties:

- if $v_c(t) \in \sigma(I(\lfloor t/\tau \rfloor \tau))$ for $t \in [k\tau, (k+1)\tau]$, then $x(t) \notin B$ in the same time interval (safety)

- if $\sigma(I(k\tau)) \neq \emptyset$, $v_c(t) \in \sigma(I(\lfloor t/\tau \rfloor \tau))$ for $t \in [k\tau, (k+1)\tau]$, and $\ell(I((k+1)\tau)) \neq \{q_m\}$, then $\sigma(I((k+1)\tau)) \neq \emptyset$ (non-deadlockingness)

- if $\tilde{\sigma} \neq \sigma$ and $\tilde{\sigma}$ satisfies the two properties above, then $\tilde{\sigma}(I(k\tau)) \subseteq \sigma(I(k\tau))$ for all $k \geq 0$ (maximal permissiveness),

where $I(k\tau)$ is a state estimate *after* the correction step of Eq. (4.6) and $I((k+1)\tau) = I^c(I^p(I(k\tau), v_c), \chi)$. □

See Fig. 4.1 for a graphical depiction of Prob. IV.1.

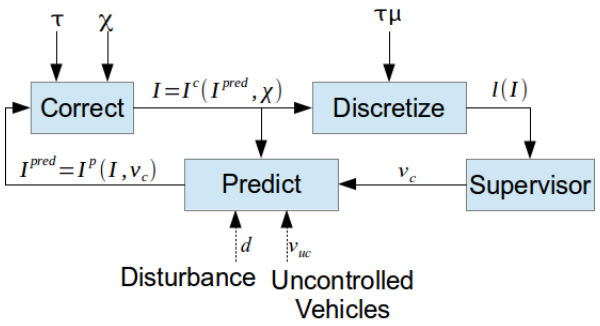


Figure 4.1: The system process in real time. At times $0, \tau, 2\tau, \ldots$, a measurement $\chi \in X$ is obtained. This measurement is used to correct the previous state estimate I^{pred} through $I = I^c(I^{pred}, \chi) = I^{pred} \cap L(\chi)$. The corrected state estimate I is then discretized to a lattice with spacing $\tau\mu$ as $\ell(I)$, and $\ell(I)$ is sent to the supervisor. The controlled vehicles then choose some control action v_c allowed by the supervisor, and this control action, along with the state estimate I and knowledge of the *bounds* on the actions of the disturbance and the uncontrolled vehicles (shown with dashed lines because they are not observed) are used to predict a new state estimate of possible positions at the next time instant that is a multiple of τ, given by $I^{pred} = I^p(I, v_c)$, where I^p is the function of Eq. (4.5).

4.4 Results from the Case of Perfect Measurement

This section provides a summary of results from the case of perfect measurement. We begin by defining the initial discrete event system (DES) abstraction G. In Sec. 4.5, we will modify this abstraction to include a finite set of measurement events to model measurement uncertainty, resulting in a new abstraction G'. We then proceed to recall results about state reductions and exact state reductions, which are types of relations between systems and abstractions, using alternating similarity as a basis. The definitions of state reductions and exact state reductions will be adapted to the case of imperfect measurement in Sec. 4.5, yielding the state estimate reduction and exact state estimate reduction, which are relations between prediction-correction estimators and abstractions.

4.4.1 The Initial Abstraction

Definition IV.2 (Discrete Event System Automaton)**.** A (deterministic) discrete event system is a tuple $G = (Q, E, \psi, q_0, Q_m)$ where Q is a set of states, E is a set of events, $\psi : Q \times E \to Q$ is a partial transition function, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is a set of marked states indicating the completion of some behavior of interest. □

We model the discrete time system of Eq. (4.2) by constructing a DES abstraction with three types of events: U_c, to model the actions of the controlled vehicles; U_{uc} to model the actions of the uncontrolled vehicles; and W, to model the effect of the disturbance. The sets U_c and U_{uc} were already described in Section 4.3. The set W consists of a set of "discretized disturbances" and is defined by $W = \{k\tau\mu : k \in \mathbb{Z} \wedge \lfloor \delta_{min}/(\tau\mu) \rfloor \leq k \leq \lceil \delta_{max}/(\tau\mu) \rceil\}^n$. It can be shown that this set W satisfies the following property: For any $q, q' \in \tilde{Q}$ and $u \in U$, there exist $x \in \ell^{-1}(q)$, $x' \in \ell^{-1}(q')$, and $\delta \in \Delta$ such that $x + u + \delta = x'$ if and only if $\exists w \in W$ such that $q + u + w = q'$. Each of the three event types constitutes one layer of a three layer DES automaton transition function ψ. In order to have a well defined DES automaton, this requires the introduction of two sets of intermediate states Q_{I1} and Q_{I2}. More specifically, we have $\psi \subseteq (\tilde{Q} \times U_c \times Q_{I1}) \cup (Q_{I1} \times U_{uc} \times Q_{I2}) \cup (Q_{I2} \times W \times \tilde{Q})$, satisfying $\psi(q, u_c u_{uc} w) = q + u + w$, for all $q \in \tilde{Q}$, $u_c \in U_c$, $u_{uc} \in U_{uc}$ and $w \in W$. See Fig. 4.2 for a depiction of the transition function ψ.

In order to allow for the possibility of multiple initial states, we define a set Q_0 of possible initial states, which we model through a dummy initial state q_0 and a set of events $E_Q := \{e_q : q \in Q_0\}$ with $\psi(q_0, e_q) = q$. The final DES abstraction is then defined as:

$$G := (Q, E, \psi, q_0, Q_m), \tag{4.7}$$

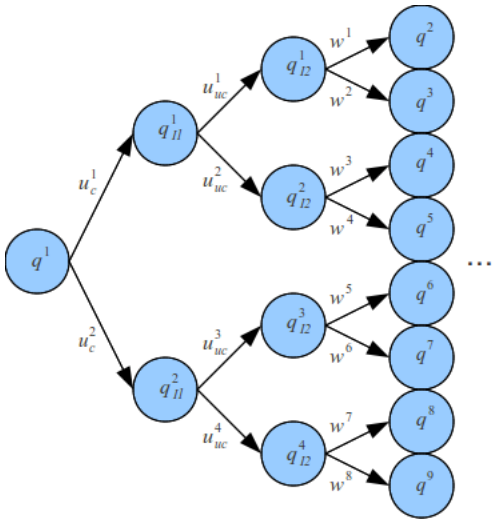


Figure 4.2: The transition function ψ of DES automaton G, consisting of three layers, one for each of the event categories U_c, U_{uc}, W, and separated by intermediate states.

where $Q = q_0 \cup \tilde{Q} \cup Q_{I1} \cup Q_{I2}$, $E = E_Q \cup U_c \cup U_{uc} \cup W$, and $Q_m = \{q_m\}$. The events in U_c are defined to be controllable, whereas the events in U_{uc} and W are defined to be uncontrollable.

4.4.2 State Reductions & Exact State Reductions

This section presents definitions and theorems about state reductions and exact state reductions found in *Dallal et al.* (2014). With the exceptions of Props. IV.11 and IV.15, the results are generic and hence the notation X and U in the following definitions need not be the same as in this work's vehicle control problem.

4.4.2.1 Preliminaries

Definition IV.3 (System, *Dallal et al.* (2014))**.** A system S is defined as a tuple $S = (X, U, \rightarrow, Y, H)$, where X is the set of states, U is a set of control inputs, $\rightarrow \subseteq X \times U \times X$ is a transition relation, Y is an output set, and $H : X \rightarrow Y$ is the output function. □

For a system $S = (X, U, \rightarrow, Y, H)$, we will use the notation $\mathbf{Post}_u(x) := \{x' \in X :$

$(x, u, x') \in \rightarrow\}$ and $U(x) := \{u \in U : \mathbf{Post}_u(x) \neq \emptyset\}$. In the remainder of this work, it will be assumed that all systems satisfy the property $H(x_1) = H(x_2) \Rightarrow U(x_1) = U(x_2)$, for all $x_1, x_2 \in X$. In words, this means that any two states with the same output should not be distinguishable by their available set of inputs. In the context of the vehicle control application, the system is defined by Eqs. (4.2) and (4.3).

Definition IV.4 (Run, *Dallal et al.* (2014))**.** A run ρ of length n for a system $S = (X, U, \rightarrow, Y, H)$ is a sequence of past states and inputs $(x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n)$, such that $u^i \in U(x^i)$ and $x^{i+1} \in \mathbf{Post}_{u^i}(x^i)$ for $i = 0, \ldots, n-1$. The set of runs of length n is denoted by $R_n(S)$ and the set of runs is $R(S) = \bigcup_{i=0}^{\infty} R_n(S)$. Given run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n)$, we define the notation $tgt(\rho) := x^n$ and $\rho(k) := (x^0, u^0, \ldots, x^{k-1}, u^{k-1}, x^k)$, called a *prefix* of ρ. □

Definition IV.5 (History, *Dallal et al.* (2014))**.** A history θ of length n for a system $S = (X, U, \rightarrow, Y, H)$ is a sequence of past outputs and inputs $(y^0, u^0, \ldots, y^{n-1}, u^{n-1}, y^n)$, such that there exists a run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n) \in R_n(S)$ that is consistent with θ, in the sense that $y^i = H(x^i)$ for $i = 0, \ldots, n$. The set of histories of length n is denoted by $\Theta_n(S)$ and the set of histories is $\Theta(S) = \bigcup_{i=0}^{\infty} \Theta_n$. We will also write $\theta(\rho)$ to mean the unique history produced by a run $\rho \in R$. Given history $\theta = (y^0, u^0, \ldots, y^{n-1}, u^{n-1}, y^n)$, we define the notation $\theta(k) := (y^0, u^0, \ldots, y^{k-1}, u^{k-1}, y^k)$ and $tgt(\theta) := y^n$, as was the case with runs. □

Definition IV.6 (Supervisor, *Dallal et al.* (2014))**.** A supervisor σ for a system $S = (X, U, \rightarrow, Y, H)$ is a function $\sigma : \Theta \rightarrow 2^U$ that chooses which control inputs to enable/disable after each history. A supervisor is called memoryless if it is of the form $\sigma : Y \rightarrow 2^U$. A run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n) \in R_n(S)$ is allowed by supervisor σ if $u^i \in \sigma(\theta(\rho(i)))$, for $i = 0, \ldots, n-1$. □

Definition IV.7 (Specification, *Dallal et al.* (2014))**.** A safety specification for a system $S = (X, U, \rightarrow, Y, H)$ is a subset $Safe \subseteq \rightarrow$ of transitions that we would like

the system S to be restricted to. A marking specification for S is a set $X_m \subseteq X$ of "special" or marked states. We say that S is deadlocking if there exists a run ρ such that $U(tgt(\rho)) = \emptyset$ and $tgt(\rho) \notin X_m$. □

4.4.2.2 The State Reduction

Definition IV.8 (State Reduction, *Dallal et al.* (2014))**.** Given two systems S_a and S_b with $Y_a = Y_b := Y$, we say that S_a is a state reduction of S_b with state relation $R \subseteq X_a \times X_b$ and output dependent control relation $C : Y \to 2^{U_a \times U_b}$ (hereafter referred to only as control relation) if:

1. $R^{-1} = \{(x_b, x_a) \subseteq X_b \times X_a : (x_a, x_b) \in R\}$ is a function.
2. For every $y \in Y$, the relation $C(y) \subseteq U_a \times U_b$ is a bijection relation.
3. $H_a(x_a) = H_b(x_b)$ if and only if $(x_a, x_b) \in R$.
4. $\forall (x_a, u_a, x'_a) \in \to_a$, $\exists (x_b, u_b, x'_b) \in \to_b$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(H_a(x_a))$, and $(x'_a, x'_b) \in R$.
5. $\forall (x_b, u_b, x'_b) \in \to_b$, $\exists (x_a, u_a, x'_a) \in \to_a$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(H_b(x_b))$, and $(x'_a, x'_b) \in R$.

□

Definition IV.9 (Induced Specification, *Dallal et al.* (2014))**.** Given system S_b with state reduction S_a, along with safety and marking specifications $Safe_b \subseteq \to_b$ and

$X_{m,b} \subseteq X_b$ on system S_b, define the induced specification on S_a as follows:

$$
\begin{aligned}
& (x_a, u_a, x'_a) \in Safe_a \subseteq\!\!\rightarrow_a \\
\Leftrightarrow \quad & \left\{ \begin{array}{l} (x_b, u_b, x'_b) \in\!\!\rightarrow_b \text{ s.t.} \\ \quad (x_a, x_b) \in R \\ \quad \wedge (u_a, u_b) \in C(H_a(x_a)) \\ \quad \wedge (x'_a, x'_b) \in R \end{array} \right\} \subseteq Safe_b \qquad (4.8) \\
& X_a \in X_{m,a} \subseteq X_a \\
\Leftrightarrow \quad & \{x_b \in X_b \text{ s.t. } (x_a, x_b) \in R\} \subseteq X_{m,b} \qquad (4.9)
\end{aligned}
$$

□

Theorem IV.10. *(from Dallal et al. (2014)) Suppose that system S_a is a state reduction of system S_b with state relation R and control relation C and that we are given safety and marking specifications $Safe_b \subseteq\!\!\rightarrow_b$ and $X_{m,b} \subseteq X_b$ for system S_b. Let $Safe_a$ and $X_{m,a}$ be the corresponding induced specifications for system S_a and suppose that we have a maximally permissive, safe, and non-deadlocking supervisor $\sigma_a : Y \to 2^{U_a}$, where Y is the (common) output space. Define the supervisor $\sigma_b : Y \to 2^{U_b}$ by $u_b \in \sigma_b(y)$ iff $\exists u_a \in \sigma_a(y)$ such that $(u_a, u_b) \in C(y)$. Then σ_b is safe, non-deadlocking, and maximally permissive among supervisors of the form $\sigma_b : Y \to 2^{U_b}$.*

Proposition IV.11. *(from Dallal et al. (2014)) Define the output maps $H_X(x) := \ell(x)$, $H_{\tilde{Q}}(q) := q$, the relation $R := \{(x, q) \in X \times \tilde{Q} : \ell(x) = q\}$, and the control relation $C(q) := \{(v_c, u_c) : v_c\tau = u_c \in U_c\}$, for all $q \in \tilde{Q}$. Then DES abstraction G of Sec. 4.4.1 is a state reduction of system* (4.2).

4.4.2.3 The Exact State Reduction

Definition IV.12 (Exact State Reduction, *Dallal et al.* (2014))**.** Given two systems S_a and S_b with $Y_a = Y_b = Y$, we say that S_a is an exact state reduction of S_b with state relation $R \subseteq X_a \times X_b$ and control relation $C : Y \to 2^{U_a \times U_b}$ if S_a is a state reduction of S_b with state and control relations R and C and:

6. $\forall (x_a, u_a, x'_a) \in \rightarrow_a, \forall x'_b \in X_b : (x'_a, x'_b) \in R, \exists (x_b, u_b, x'_b) \in \rightarrow_b$ such that $(x_a, x_b) \in R$ and $(u_a, u_b) \in C(H_a(x_a))$. □

Lemma IV.13. *(from Dallal et al. (2014)) Suppose that system S_b has an exact state reduction S_a. Then, for any history θ_b for system S_b and any $x_b \in X_b$ such that $H(x_b) = tgt(\theta_b)$, there exists a run ρ_b such that $\theta_b = \theta(\rho_b)$ and $x_b = tgt(\rho_b)$.*

In words, the above lemma implies that, when there exists an exact state reduction for system S_b, a history θ_b gives no more information about the current state of S_b than does the last output $tgt(\theta_b)$. The following theorem follows immediately from this observation.

Theorem IV.14. *(from Dallal et al. (2014)) Suppose that system S_a is an exact state reduction (2) of system S_b and that all other conditions of Thm. IV.10 are satisfied. Then the obtained supervisor σ_b will be safe, non-deadlocking, and maximally permissive among supervisors of the form $\sigma_b : \Theta \to 2^{U_b}$.*

Proposition IV.15. *(from Dallal et al. (2014)) Define $H_X(\cdot)$, $H_{\tilde{Q}}(\cdot)$, R, and C as in Prop. IV.11. If δ_{min} and δ_{max} are both integer multiples of $\tau\mu$, then DES abstraction G of Sec. 4.4.1 is an exact state reduction of system* (4.2).

4.5 Observers and State Estimation

In this section, we show how to modify the DES abstraction G of Sec. 4.4.1 to deal with measurement uncertainty. Specifically, we accomplish this by obtaining

a finite set of equivalence classes for the measurements, given the discretization of Eq. (4.3). This set of equivalence classes will then be used as a fourth type of event in our DES abstraction, resulting in a modified DES abstraction G'. We then proceed to define two new relations between estimators of partially observed systems and abstractions, the state estimate reduction and exact state estimate reduction, which are modified versions of the state reduction and exact state reduction. We then show that analogous versions of Thms. IV.10 and IV.14 hold with these two new relations.

4.5.1 The modified abstraction

Given the discretization function $\ell : X \to \tilde{Q}$ of Eq. (4.3) and the observation function $L : X \to 2^X$ of Eq. (4.4), define the equivalence relation $\equiv_o \subseteq X \times X$ over measurements by:

$$\chi_1 \equiv_o \chi_2 \Leftrightarrow \ell(L(\chi_1)) = \ell(L(\chi_2)). \tag{4.10}$$

For any $\chi \in X$, let $[\chi]$ denote the equivalence class of χ under relation $\equiv_o$ and let Λ denote the set of equivalence classes. These equivalence classes will constitute the set of discrete measurement events. Now define the discrete observation function $L^\Lambda : \Lambda \to 2^{\tilde{Q}}$ by:

$$L^\Lambda(\lambda) = \{q \in \tilde{Q} : (\exists \chi \in X : [\chi] = \lambda)[L(\chi) \cap \ell^{-1}(q) \neq \emptyset]\}, \tag{4.11}$$

for any $\lambda \in \Lambda$.

We now define $\psi^c : \tilde{Q} \times \Lambda \to \tilde{Q}$, the analogue of the correction operator of Eq. (4.17) in the DES domain:

$$\psi^c(q, \lambda) = \begin{cases} q, & \text{if } q \in L^\Lambda(\lambda) \\ \text{undefined}, & \text{else} \end{cases} \tag{4.12}$$

In words, $\psi^c(q, \lambda)$ is defined if there exists measurement χ satisfying $[\chi] = \lambda$ and state

x satisfying $\ell(x) = q$ such that x is consistent with measurement χ.

We then use ψ^c as a fourth layer in the transition function ψ of DES abstraction G defined in Sec. 4.4.1 in order to obtain a new DES abstraction G'. Whereas the language of G satisfied $\mathcal{L}(G) \subseteq E_Q(U_c U_{uc} W)^*$, the language of G' should satisfy $\mathcal{L}(G') \subseteq E_Q(\Lambda U_c U_{uc} W)^*$. We therefore create a new layer of states $\tilde{Q}'$ that is a copy of $\tilde{Q}$ and precedes $\tilde{Q}$. This yields a set of states $Q' = Q \cup \tilde{Q}'$, a set of events $E' = E \cup \Lambda$, and a transition function $\psi' : Q' \times E' \to Q'$ defined by:

$$\psi'(q,e) = \begin{cases} q' \in \tilde{Q}', & \text{if } q = q_0 \wedge e = e_{q'} \in E_Q \\ \psi^c(q,e), & \text{if } q \in \tilde{Q}' \wedge e \in \Lambda \\ \psi(q,e), & \text{else} \end{cases} \tag{4.13}$$

The DES automaton G' is defined as $G' := (Q', E', \psi', q_0, Q_m)$. Finally, we take the events of U_{uc} and W to be uncontrollable and unobservable, the events of U_c to be observable and controllable, and the events of Λ to be observable but uncontrollable.

Now consider the observer $Obs(G')$ of this modified abstraction, in the terminology of supervisory control theory of DES (see Sec. 2.5.2 of *Cassandras and Lafortune* (2008)). Let $E_o = U_c \cup \Lambda$ and $E_{uo} = E_Q \cup U_{uc} \cup W$ be the sets of observable and unobservable events of G', respectively. Define $UR : Q' \to 2^{Q'}$ by $UR(q) = \{q' \in Q' : \exists s \in E_{uo}^* \text{ s.t. } q' = \psi'(q,s)\}$ (called the *Unobservable Reach*) and extend this definition to sets by $UR(\iota) = \cup_{q \in \iota} UR(q)$, for any $\iota \subseteq Q'$. Also define $OR : 2^{Q'} \times E_o \to 2^{Q'}$ by $OR(\iota, e) = \cup_{q \in \iota} \psi'(q,e)$. Then $Obs(G')$ is defined as the accessible part of the automaton $(2^{Q'}, E_o, \overline{\psi}, UR(q_0), \iota_m)$, where $\overline{\psi}$ and ι_m are defined by

$$\begin{aligned} \overline{\psi}(\iota, e_o) &= \begin{cases} \text{undefined}, & OR(\iota, e_o) = \emptyset \\ UR(OR(\iota, e_o)), & \text{else} \end{cases} \\ \iota_m &= \{\iota \subseteq Q_m\} \end{aligned}$$

The observer $Obs(G')$ is therefore defined over state estimates of G'. Furthermore, for any string $s \in (\Lambda U_c)^*$, $\overline{\psi}(UR(q_0), s)$ will consist of the set of states of G' that are consistent with the sequence s of measurements and control actions. More specifically, we can define prediction and correction functions $\overline{\psi}^p : 2^{\tilde{Q}} \times U_c \to 2^{\tilde{Q}}$ and $\overline{\psi}^c : 2^{\tilde{Q}} \times \Lambda \to 2^{\tilde{Q}}$ similar to I^p and I^c of Eqs. (4.5) and (4.6), but for state estimates $\iota \subseteq \tilde{Q}$:

$$\overline{\psi}^p(\iota, u_c) = \bigcup_{q \in \iota} \bigcup_{u_{uc} \in U_{uc}} \bigcup_{w \in W} \psi(q, u_c u_{uc} w) \tag{4.14}$$

$$\overline{\psi}^c(\iota, \lambda) = \iota \cap L^{\Lambda}(\lambda) \tag{4.15}$$

With these definitions, it can be shown that $\overline{\psi}$ is closely related to $\overline{\psi}^p$ and $\overline{\psi}^c$. We will return to this issue in Sec. 4.6.1.3.

It should be noted that the above definition of the set of marked states ι_m of the observer is not standard. The usual definition would be $\iota_m = \{\iota \subseteq Q' : \iota \cap Q_m \neq \emptyset\}$. We use $\iota_m = \{\iota \subseteq Q_m\}$ here because we will use this observer in a supervisory control problem and we wish to be certain that all vehicles have crossed the intersection when the observer reaches a marked state.

4.5.2 State Estimate Reductions & Exact State Estimate Reductions

As was the case in Sec. 4.4.2, the results of this section are generic and hence the notation X and U in the following definitions need not be the same as in this work's vehicle control problem.

Definition IV.16 (Partially Observed System)**.** A partially observer system is a tuple $S = (X, U, \to, Y, H, O, L)$, where the first five elements have the same interpretation as in Def. IV.3, O is a set of observations, and $L : O \to 2^X$ defines the set of states that are consistent with any given observation. □

In the above definition, we differentiate between outputs and observations. Outputs should be thought of as states corresponding to a discretization, whereas ob-

servations should be thought of as physical measurements. For every state of the system, there is a single output, but many possible consistent observations. In the context of the vehicle control application, the partially observed system is defined by Eqs. (4.2)-(4.4).

Definition IV.17 (Estimator System)**.** Given a partially observed system $S = (X, U, \rightarrow, Y, H, O, L)$, define the prediction and correction functions $I^p : 2^X \times U \rightarrow 2^X$ and $I^c : 2^X \times O \rightarrow 2^X$ by

$$I^p(I, u) = \bigcup_{x \in I} \{x' \in X : (x, u, x') \in \rightarrow\} \tag{4.16}$$

$$I^c(I, o) = I \cap L(o) \tag{4.17}$$

With these definitions, we denote S's estimator system by $\overline{S} = (I(S), U, \Rightarrow, 2^Y, H)$ where

$$I(S) = \{I' \in 2^X \setminus \{\emptyset\} : (\exists I \subseteq X)(\exists o \in O) \text{ s.t. } I' = I^c(I, o)\} \tag{4.18}$$

and, for any $(I, u, I') \in I(S) \times U \times 2^X \setminus \{\emptyset\}$,

$$(I, u, I') \in \Rightarrow \Leftrightarrow \exists o \in O \text{ s.t. } I' = I^c(I^p(I, u), o). \tag{4.19}$$

Note that, from Eqs. (4.18) and (4.19), it will indeed be the case that $(I, u, I') \in \Rightarrow$ yields $I' \in I(S)$, so that we do in fact have that $\Rightarrow \subseteq I(S) \times U \times I(S)$. □

In the above definition, we use $I(S)$ rather than simply 2^X because the estimator system is meant to model only state estimates reached after correction steps (as in Eq. (4.17)), and not those reached after prediction steps (as in Eq. (4.16)). This is because we will again define a supervisor for the estimator system and new control actions occur only after observations.

Remark IV.18 (Systems Defined over State Estimates). In the definitions that follow, we will refer to "systems defined over state estimates". These are systems of the

form: $\overline{S} = (I(S), U, \overline{\Rightarrow}, 2^Y, H)$, where $I(S) \subseteq 2^X$ for some set X. Thus, any estimator system is a system defined over state estimates. However, a system defined over state estimates need not be the estimator system of any partially observed system. In particular, a system defined over state estimates can be obtained directly by abstraction of some estimator system (or of any other system defined over state estimates). In order to avoid differentiating between the cases where $\overline{S}$ is or is not the estimator of some partially observed system S, we will abuse notation and use $I(S)$ in both cases as the set of state estimates over which $\overline{S}$ is defined.

Definition IV.19 (State Estimate Reduction)**.** Given partially observed system S_b with corresponding estimator system $\overline{S}_b$ and system $\overline{S}_a$ defined over state estimates and sharing the same set of outputs 2^Y, we say that $\overline{S}_a$ is a state estimate reduction of $\overline{S}_b$ with state relation $\overline{R} \subseteq 2^{X_a} \times 2^{X_b}$ and control relation $\overline{C} : H_a(I(S_a)) \cup H_b(I(S_b)) \to 2^{U_a \times U_b}$ if:

0. $H_a(I(S_a)) = H_b(I(S_b)) := \overline{Y}$.

1. $\overline{R}^{-1}$ is a function.

2. For every $\overline{y} \in \overline{Y}$, the relation $\overline{C}(\overline{y}) \subseteq U_a \times U_b$ is a bijection.

3. $H_a(I_a) = H_b(I_b)$ if and only if $(I_a, I_b) \in \overline{R}$.

4. $\forall (I_a, u_a, I'_a) \in \overline{\Rightarrow}_a$, $\exists (I_b, u_b, I'_b) \in \overline{\Rightarrow}_b$ such that $(I_a, I_b) \in \overline{R}$, $(u_a, u_b) \in \overline{C}(H_a(I_a))$, and $(I'_a, I'_b) \in \overline{R}$.

5. $\forall (I_b, u_b, I'_b) \in \overline{\Rightarrow}_b$, $\exists (I_a, u_a, I'_a) \in \overline{\Rightarrow}_a$ such that $(I_a, I_b) \in \overline{R}$, $(u_a, u_b) \in \overline{C}(H_b(I_b))$, and $(I''_a, I'_b) \in \overline{R}$, for some $I''_a \subseteq I'_a$. □

This is almost identical to a state reduction, with the main difference being property 5. Given two transitions $(I_{b,1}, u_b, I'_{b,1})$ and $(I_{b,2}, u_b, I'_{b,2})$, both in $\overline{\Rightarrow}_b$ with $H_b(I_{b,1}) = H_b(I_{b,2})$ but $H_b(I'_{b,1}) \subseteq H_b(I'_{b,2})$, the state estimate reduction requires the

existence of some $(I_{a,2}, u_a, I'_{a,2}) \in \Rightarrow_a$ satisfying $(I_{a,2}, I_{b,2}) \in \overline{R}$, $(u_a, u_b) \in \overline{C}(H_b(I_{b,2}))$, and $(I'_{a,2}, I'_{b,2}) \in \overline{R}$, but *does not* require the existence of some $(I_{a,1}, u_a, I'_{a,1}) \in \Rightarrow_a$ satisfying $(I_{a,1}, I_{b,1}) \in \overline{R}$, $(u_a, u_b) \in \overline{C}(H_b(I_{b,1}))$, and $(I'_{a,1}, I'_{b,1}) \in \overline{R}$. The intuition for this definition is as follows. A memoryless supervisor σ (for either the a or b system) with domain $\overline{Y}$ must satisfy $\sigma(H(I_1)) = \sigma(H(I_2))$ if $H(I_1) = H(I_2)$. If $H(I'_1) \subseteq H(I'_2)$ then we will logically have $\sigma(H(I'_1)) \supseteq \sigma(H(I'_2))$, since a supervisor cannot be more permissive with less information. Hence, the existence of transition (I_1, u, I'_1) will not place any further restrictions on $\sigma(H(I_1)) = \sigma(H(I_2))$ than those restrictions placed by the existence of transition (I_2, u, I'_2).

Definition IV.20 (Specifications for Partially Observed Systems)**.** Given a partially observed system S with corresponding estimator system $\overline{S}$, along with safety and marking specifications $Safe \subseteq \rightarrow$ and $X_m \subseteq X$, define the safety and marking specifications $\overline{Safe}$ and I_m on $\overline{S}$ as follows:

$$
\begin{aligned}
& (I, u, I') \in \overline{Safe} \subseteq \Rightarrow \\
\Leftrightarrow \quad & \{(x, u, x') \in \rightarrow \text{ s.t. } x \in I \wedge x' \in I'\} \subseteq Safe && (4.20) \\
& I \in I_m \in I(S) \Leftrightarrow I \subseteq X_m && (4.21)
\end{aligned}
$$

□

Definition IV.21 (Induced Specification for Partially Observed Systems)**.** Given partially observed system S_b with corresponding estimator system $\overline{S}_b$ and system $\overline{S}_a$ defined over state estimates such that $\overline{S}_a$ is a state estimate reduction of system $\overline{S}_b$, along with safety and marking specifications $\overline{Safe}_b$ and $I_{m,b}$ on $\overline{S}_b$, the safety and

marking specifications $\overline{Safe}_a$ and $I_{m,a}$ on system $\overline{S}_a$ are obtained as follows:

$$
\begin{aligned}
& (I_a, u_a, I'_a) \in \overline{Safe}_a \subseteq \rightrightarrows_a \\
\Leftrightarrow \quad & \left\{ \begin{array}{l} (I_b, u_b, I'_b) \in \rightrightarrows_b \text{ s.t.} \\ \quad (I_a, I_b) \in \overline{R} \\ \quad \wedge (u_a, u_b) \in C(H_a(I_a)) \\ \quad \wedge \exists I''_a \subseteq I'_a : (I''_a, I'_b) \in \overline{R} \end{array} \right\} \subseteq \overline{Safe}_b \qquad (4.22) \\
& I_a \in I_{m,a} \in I(S_a) \\
\Leftrightarrow \quad & \{I_b \in I(S_b) \text{ s.t. } (I_a, I_b) \in \overline{R}\} \subseteq I_{m,b} \qquad (4.23)
\end{aligned}
$$

□

This definition is almost precisely analogous to Def. IV.9, with the exception that we must use $\exists I''_a \subseteq I'_a : (I''_a, I'_b) \in \overline{R}$ rather than simply $(I'_a, I'_b) \in \overline{R}$ in Eq. (4.22). This is due to Property 5 of Def. IV.19.

Theorem IV.22. *Given partially observed system S_b with corresponding estimator system $\overline{S}_b$ and system $\overline{S}_a$ defined over state estimates, suppose that $\overline{S}_a$ is a state estimate reduction of system $\overline{S}_b$ with state relation $\overline{R}$ and control relation $\overline{C}$. Suppose that both systems S_a and S_b satisfy the property that $x_1, x_2 \in I \in I(S) \Rightarrow U(x_1) = U(x_2)$ and that we are given safety and marking specifications $Safe_b \subseteq \rightarrow_b$ and $X_{m,b} \subseteq X_b$ for system S_b. Let $\overline{Safe}_a$ and $I_{m,a}$ be the corresponding induced specifications for system $\overline{S}_a$ and suppose that we have a maximally permissive, safe, and non-deadlocking supervisor $\sigma_a : \overline{Y} \to 2^{U_a}$, where Y is the (common) output space. Define the supervisor $\sigma_b : \overline{Y} \to 2^{U_b}$ by $u_b \in \sigma_b(\overline{y})$ iff $\exists u_a \in \sigma_a(\overline{y})$ such that $(u_a, u_b) \in C(\overline{y})$. Then σ_b is safe, non-deadlocking, and maximally permissive among supervisors of the form $\sigma_b : \overline{Y} \to 2^{U_b}$.*

Proof. We proceed in three claims.

Claim 1: $(I_a, I_b) \in \overline{R} \wedge (\sigma_a(H_a(I_a)) \neq \emptyset \vee I_a \in I_{m,a}) \Rightarrow (\sigma_b(H_b(I_b)) \neq \emptyset \vee I_b \in I_{m,b})$.

By definition of $I_{m,a}$, $I_a \in I_{m,a} \Leftrightarrow I_b \in I_{m,b}$, for all $I_b : (I_a, I_b) \in \overline{R}$, so that $(I_a, I_b) \in \overline{R} \wedge I_a \in I_{m,a} \Rightarrow I_b \in I_{m,b}$. By definition of σ_b, $u_b \in \sigma_b(\overline{y})$ iff $\exists u_a \in \sigma_a(\overline{y})$ such that $(u_a, u_b) \in \overline{C}(\overline{y})$. But $(I_a, I_b) \in \overline{R} \Rightarrow H_a(I_a) = H_b(I_b)$, so that $(I_a, I_b) \in \overline{R} \wedge \sigma_a(H_a(I_a)) \neq \emptyset \Rightarrow \sigma_b(H_b(I_b)) \neq \emptyset$.

By assumption, $x_{b,1}, x_{b,2} \in I_b \Rightarrow U_b(x_{b,1}) = U_b(x_{b,2})$, from which it follows that, if $\sigma_b(H_b(I_b)) \neq \emptyset$, then any $x_b \in I_b$ is not deadlocked under σ_b.

Claim 2: $\forall I_b \in I(S_b)$, $u_b \in \sigma_b(H_b(I_b)) \Rightarrow \forall I'_b \in \mathbf{Post}_{u_b}(I_b), (I_b, u_b, I'_b) \in \overline{Safe}_b \wedge [\sigma_b(H_b(I'_b)) \neq \emptyset \vee I'_b \in I_{m,b}]$.

Consider any $I_b \in I(S_b)$ and any $u_b \in \sigma_b(H_b(I_b))$. By property (1) of Def. IV.19, $\overline{R}^{-1}$ is a function. Therefore let I_a be the unique member of $I(S_a)$ such that $(I_a, I_b) \in \overline{R}$. By property (3) of Def. IV.19, $H_a(I_a) = H_b(I_b) = \overline{y}$ for some $\overline{y} \in \overline{Y}$. By property (2) of Def. IV.19, $\overline{C}(\overline{y})$ is a bijection. Therefore let u_a be the unique member of U_a such that $(u_a, u_b) \in \overline{C}(\overline{y})$. From the definition of σ_b, it follows that $u_a \in \sigma_a(H_a(I_a))$. Thus, $\forall I'_a \in \mathbf{Post}_{u_a}(I_a), (I_a, u_a, I'_a) \in \overline{Safe}_a \wedge [\sigma_a(H_a(I'_a)) \neq \emptyset \vee I'_a \in I_{m,a}]$. From the way that $\overline{Safe}_a$ was defined, this implies that $(I_b, u_b, I'_b) \in \overline{Safe}_b$, for all $I'_b \in \mathbf{Post}_{u_b}(I_b)$ such that there exist $I'_a \in \mathbf{Post}_{u_a}(I_a)$ and $I''_a \subseteq I'_a$ with $(I''_a, I'_b) \in \overline{R}$. But property (5) of Def. IV.19 states that $\forall I'_b \in \mathbf{Post}_{u_b}(I_b)$, $\exists I'_a \in \mathbf{Post}_{u_a}(I_a)$ satisfying $(I''_a, I'_b) \in \overline{R}$, for some $I''_a \subseteq I'_a$. From this and the previous statement, it follows that $(I_b, u_b, I'_b) \in \overline{Safe}_b$, for all $I'_b \in \mathbf{Post}_{u_b}(I_b)$. It similarly follows from property (5) of Def. IV.19 along with Claim 1 that $\sigma_b(H_b(I'_b)) \neq \emptyset \vee I'_b \in I_{m,b}$, for all $I'_b \in \mathbf{Post}_{u_b}(I_b)$.

Thus σ_b is safe and non-deadlocking. Given any supervisor $\sigma'_b : \overline{Y} \rightarrow 2^{U_b}$, let $\sigma'_a : \overline{Y} \rightarrow 2^{U_a}$ be defined by $u_a \in \sigma'_a(\overline{y})$ iff $\exists u_b \in \sigma'_b(\overline{y})$ such that $(u_a, u_b) \in \overline{C}(\overline{y})$ and let the function $\sigma_{b \rightarrow a}$ be the mapping which takes a supervisor σ'_b for system b to the supervisor σ'_a for system a in this way.

Claim 3: If σ'_b is safe and non-deadlocking then so is $\sigma'_a = \sigma_{b \rightarrow a}(\sigma'_b)$.

Suppose that σ'_b is safe and non-deadlocking and take $\sigma'_a = \sigma_{b \rightarrow a}(\sigma'_b)$. Consider any $I_a \in I(S_a)$, any $u_a \in \sigma'_a(H_a(I_a))$, and any $I'_a \in \mathbf{Post}_{u_a}(I_a)$. By property (4)

of Def. IV.19, there exists $(I_{b,1}, u_b, I'_{b,1}) \in \Rightarrow_b$ such that $(I_a, I_{b,1}) \in \overline{R}$, $(u_a, u_b) \in \overline{C}(H_a(I_a))$, and $(I'_a, I'_{b,1}) \in \overline{R}$. From the definition of $\sigma_{b\to a}(\sigma'_b)$, it must be that $u_b \in \sigma'_b(H_b(I_{b,1}))$. By property (3) of Def. IV.19, any $I_{b,2}$ such that $(I_a, I_{b,2}) \in \overline{R}$ must satisfy $H_b(I_{b,1}) = H_a(I_a) = H_b(I_{b,2})$. It follows that $u_b \in \sigma'_b(H_b(I_b))$, for *any* $I_b \in I(S_b)$ such that $(I_a, I_b) \in \overline{R}$. Since σ'_b is safe and non-deadlocking, it follows that, for all $(I_b, u_b, I'_b) \in \Rightarrow_b$ such that $(I_a, I_b) \in \overline{R}$, we have that $(I_b, u_b, I'_b) \in \overline{Safe}_b \wedge [\sigma'_b(H_b(I'_b)) \neq \emptyset \vee I'_b \in I_{m,b}]$. It follows from the definition of $\overline{Safe}_a$ that $(I_a, u_a, I'_a) \in \overline{Safe}_a$ and from the definitions of $\sigma_{b\to a}(\sigma'_b)$ and $I_{m,a}$ that $\sigma'_a(H_a(I'_a)) \neq \emptyset \vee I'_a \in I_{m,a}$. Since $I'_a \in \mathbf{Post}_{u_a}(I_a)$ was arbitrary, it follows that u_a is a safe and non-deadlocking control decision from I_a. Since $I_a \in I(S_a)$ and $u_a \in \sigma'_a(H_a(I_a))$ were arbitrary, it follows that $\sigma'_a = \sigma_{b\to a}(\sigma'_b)$ is safe and non-deadlocking.

It is obvious from the definition of $\sigma_{b\to a}$ that $\sigma'_b \supseteq \sigma_b \Rightarrow \sigma_{b\to a}(\sigma'_b) \supseteq \sigma_{b\to a}(\sigma_b)$. Thus, if there exists a safe and non-deadlocking supervisor $\sigma'_b \supseteq \sigma_b$ then it follows that σ_a is not maximally permissive, a contradiction. □

We next define the exact state estimate reduction. As in the case of the state estimate reduction, the definition of the exact state estimate reduction is not directly analogous to the corresponding definition in the perfectly measured case, for reasons that will be explained shortly. We begin by defining a notion of "reachable" state estimates and work with those in the definition of an exact state estimate reduction.

Definition IV.23 (Reachable State Estimates)**.** Given a system defined over state estimates $\overline{S}$, let $I'(S) \subseteq I(S)$ be some subset of state estimates, hereafter referred to as the set of reachable state estimates and let $\Rightarrow'_b$ be the transition function $\Rightarrow_b$, restricted to the domain of such states. Mathematically,

$$\Rightarrow'_b = \Rightarrow_b \cap I'(S) \times U_b \times I'(S) \tag{4.24}$$

□

Definition IV.24 (Exact State Estimate Reduction)**.** Given partially observed system S_b with corresponding estimator system $\overline{S}_b$ and system $\overline{S}_a$ defined over state estimates and sharing the same set of outputs 2^Y, we say that $\overline{S}_a$ is an exact state estimate reduction of $\overline{S}_b$ with state relation $\overline{R} \subseteq 2^{X_a} \times 2^{X_b}$, control relation $\overline{C} : \overline{Y} \to 2^{U_a \times U_b}$, and reachable set of state estimates $I'(S_b)$ if $\overline{S}_a$ is a state estimate reduction of $\overline{S}_b$ with state and control relations $\overline{R}$ and $\overline{C}$ and:

5. $\forall (I_b, u_b, I'_b) \in \rightrightarrows_b, \exists (I_a, u_a, I'_a) \in \rightrightarrows_a$ such that $(I_a, I_b) \in \overline{R}$, $(u_a, u_b) \in \overline{C}(H_b(I_b))$, and $(I'_a, I'_b) \in \overline{R}$.

6. $\forall (I_a, u_a, I'_a) \in \rightrightarrows_a$, $[\forall I'_b \in I'(S_b) : (I'_a, I'_b) \in \overline{R}]$, $\exists (I_b, u_b, I'_b) \in \rightrightarrows'_b$ such that $(I_a, I_b) \in \overline{R}$, and $(u_a, u_b) \in \overline{C}(H_a(I_a))$.

7. $\forall \overline{y} \in \overline{Y}$, $\forall x_b \in X_b : H_b(x_b) \in \overline{y}$, $\exists I_b \in I'(S_b)$ such that $H_b(I_b) = \overline{y}$ and $x_b \in I_b$.

Note that Property 5 here replaces (strengthens) Property 5 of Def. IV.19. □

Thus, the exact state estimate reduction (unlike the state estimate reduction) uses the direct analogue of Property 5 of Def. IV.8. Property 6 of this definition, however, is not directly analogous to Property 6 of Def. IV.12. The analogous condition would require the existence of some $(I_b, u_b, I'_b) \in \rightrightarrows_b$, for every $I'_b \subseteq X_b$ such that $(I'_a, I'_b) \in \overline{R}$. This would constitute an unreasonable requirement since, in particular, it would require reachability of state estimates that are not connected sets or even of "stranger" sets like the set of points in a region whose coordinates are all irrational. What is required here is a property that would yield the equivalent of Lemma IV.13. Lemma IV.25 below shows that properties 6 and 7 above suffice.

Lemma IV.25. *Given partially observed system S_b with corresponding estimator system $\overline{S}_b$ and system $\overline{S}_a$ defined over state estimates, suppose that $\overline{S}_a$ is an exact state estimate reduction of system $\overline{S}_b$ with state relation $\overline{R}$ and control relation $\overline{C}$. Then, for any history $\overline{\theta}_b$ for system $\overline{S}_b$ and any $x_b \in X_b$ such that $H_b(x_b) \in tgt(\overline{\theta}_b)$, there exists a run $\overline{\rho}_b$ such that $\overline{\theta}_b = \theta(\overline{\rho}_b)$ and $x_b \in tgt(\overline{\rho}_b)$.*

Proof. We first claim that, for any history $\overline{\theta}_b \in \Theta(\overline{S}_b)$ and any $I'_b \in I'(S_b)$ such that $H_b(I'_b) = tgt(\overline{\theta}_b)$, there exists a run $\overline{\rho}_b$ such that $\theta(\overline{\rho}_b) = \overline{\theta}_b$ and $tgt(\overline{\rho}_b) = I'_b$. We proceed by induction. The base case is trivially true, so assume that the claim holds up to histories of length k and consider a pair of histories $\overline{\theta}_b \in \Theta_k(\overline{S}_b)$ and $\overline{\theta}'_b \in \Theta_{k+1}(\overline{S}_b)$ such that $\overline{\theta}_b$ is a prefix of $\overline{\theta}'_b$. Let $\overline{\rho}'_b = (I_b^0, \ldots, I_b^k, u_b^k, I_b^{k+1}) \in R_{k+1}(\overline{S}_b)$ be such that $\overline{\theta}'_b = \theta(\overline{\rho}'_b)$ and consider any $I'_b \in I'(S_b)$ such that $H_b(I'_b) = tgt(\overline{\theta}'_b)$. Since $(I_b^k, u_b^k, I_b^{k+1}) \in \rightrightarrows_b$, it follows from Property 5 that there exists $(I_a^k, u_a^k, I_a^{k+1}) \in \rightrightarrows_a$ such that $(I_a^k, I_b^k) \in \overline{R}$, $(u_a^k, u_b^k) \in \overline{C}(H_b(I_b^k))$, and $(I_a^{k+1}, I_b^{k+1}) \in \overline{R}$. By Property 3, $H_b(I'_b) = H_b(I_b^{k+1}) = H_a(I_a^{k+1})$ and hence $(I_a^{k+1}, I'_b) \in \overline{R}$. It therefore follows by Property 6 that there exists $(I_b, u_b, I'_b) \in \rightrightarrows'_b$ such that $(I_a^k, I_b) \in \overline{R}$, and $(u_a^k, u_b) \in \overline{C}(H_a(I_a^k))$. Since $(I_a^k, I_b^k) \in \overline{R}$ and $(I_a^k, I_b) \in \overline{R}$, we have by Property 3 that $H_a(I_a^k) = H_b(I_b^k) = H_b(I_b)$ and hence that $\overline{C}(H_a(I_a^k)) = \overline{C}(H_b(I_b^k))$. By Property 2, $\overline{C}(\cdot)$ is a bijection and hence it follows from $(u_a^k, u_b^k) \in \overline{C}(H_b(I_b^k))$ and $(u_a^k, u_b) \in \overline{C}(H_a(I_a^k))$ that $u_b = u_b^k$. By the induction hypothesis, there exists some run $\overline{\rho}_b$ such that $\theta(\overline{\rho}_b) = \overline{\theta}_b$ and $tgt(\overline{\rho}_b) = I_b$. Then the run $\overline{\rho}_b.u_b.I'_b$ satisfies $\theta(\overline{\rho}_b.u_b.I'_b) = \overline{\theta}'_b$ and $tgt(\overline{\rho}_b.u_b.I'_b) = I'_b$, which completes the proof of the claim.

Now consider any history $\overline{\theta}_b$ for system $\overline{S}_b$ and any $x_b \in X_b$ such that $H(x_b) \in tgt(\overline{\theta}_b) := \overline{y}$. By Property 7, there exists some $I_b \in I'(S_b)$ such that $H_b(I_b) = \overline{y}$ and $x_b \in I_b$. By the claim, we can find $\overline{\rho}_b$ such that $\theta(\overline{\rho}_b) = \overline{\theta}_b$ and $x_b \in I_b = tgt(\overline{\rho}_b)$, which completes the proof. □

As was the case with Lemma IV.13, the above lemma implies that, when estimator system $\overline{S}_b$ has an exact state estimate reduction, a history $\overline{\theta}_b$ gives no more information about the current state of S_b than does the last output $tgt(\overline{\theta}_b)$. Once again, the following theorem follows immediately from this observation.

Theorem IV.26. *Given partially observed system S_b with corresponding estimator system $\overline{S}_b$, suppose that system $\overline{S}_a$ defined over state estimates is an exact state estimate reduction of $\overline{S}_b$ and that all other conditions of Thm. IV.22 are satisfied. Then*

the obtained supervisor σ_b will be safe, non-deadlocking, and maximally permissive among supervisors of the form $\sigma_b : \Theta(\overline{S}_b) \to 2^{U_b}$.

We next turn our attention to finding conditions under which it is possible to obtain a state estimate reduction, using a state reduction as a basis. Specifically, suppose we have a system S_b with observations given by O_b and L_b yielding partially observed system S'_b, along with a state reduction S_a of system S_b. Procedure IV.27 shows how to construct an associated partially observed system S'_a and Thm. IV.28 shows that the estimator $\overline{S}_a$ of S'_a will "almost" be a state estimate reduction of estimator $\overline{S}_b$ of S'_b.

Procedure IV.27. Given partially observed system $S'_b = (X_b, U_b, \to_b, Y, H_b, O_b, L_b)$, let S_b be the system $S_b = (X_b, U_b, \to_b, Y, H_b)$ and suppose that $S_a = (X_a, U_a, \to_a, Y, H_a)$ is a state reduction of S_b with state relation $R \subseteq X_a \times X_b$ and control relation $C : Y \to 2^{X_a} \times 2^{X_b}$. Define the partially observed system $S'_a = (X_a, U_a, \to_a, Y, H_a, O_a, L_a)$ as follows:

1. Let the equivalence relation $\equiv_{o,b} \subseteq O_b \times O_b$ be defined by $o_{b,1} \equiv_{o,b} o_{b,2} \Leftrightarrow H_b(L_b(o_{b,1})) = H_b(L_b(o_{b,2}))$. Then let $[o_b]$ denote the equivalence class of $o_b \in O_b$ under $\equiv_{o,b}$ and define O_a as the set of such equivalence classes.

2. Define $L_a : O_a \to 2^{X_a}$ by $L_a(o_a) = \{x_a \in X_a : (\exists o_b \in O_b : o_a = [o_b])(\exists x_b \in L_b(o_b) : (x_a, x_b) \in R)\}$. □

Theorem IV.28. *Given partially observed system $S'_b = (X_b, U_b, \to_b, Y, H_b, O_b, L_b)$, let S_b be the system $S_b = (X_b, U_b, \to_b, Y, H_b)$ and suppose that $S_a = (X_a, U_a, \to_a, Y, H_a)$ is a state reduction of S_b with state relation $R \subseteq X_a \times X_b$ and control relation $C : Y \to 2^{X_a} \times 2^{X_b}$. Obtain partially observed system $S'_a = (X_a, U_a, \to_a, Y, H_a, O_a, L_a)$ as in Procedure IV.27 and let $\overline{S}_a$ and $\overline{S}_b$ be the estimator systems corresponding to S'_a and S'_b, respectively. Suppose that C satisfies the property that $y_1, y_2 \in \overline{y} \Rightarrow C(y_1) =$*

$C(y_2)$, for all $\overline{y} \in H_a(I(S'_a)) \cup H_b(I(S'_b))$. Define the state and control relations $\overline{R} \subseteq 2^{X_a} \times 2^{X_b}$ and $\overline{C} : H_a(I(S'_a)) \cup H_b(I(S'_b)) \to 2^{U_a} \times 2^{U_b}$ by:

$$(I_a, I_b) \in \overline{R} \Leftrightarrow H_a(I_a) = H_b(I_b) \tag{4.25}$$

$$\overline{C}(\overline{y}) = C(y) \text{ for any } y \in \overline{y} \tag{4.26}$$

Then $\overline{S}_a$ satisfies properties 0, 1, 2, 3, and 5 of the state estimate reduction (w.r.t. $\overline{S}_b$).

Proof.

Claim 1: For any partially observed system $S = (X, U, \to, Y, H, O, L)$,

$$I(S) = \{I \subseteq X : \exists o \in O \text{ s.t. } I \subseteq L(o)\}. \tag{4.27}$$

From Eq. (4.18), we have that $I \in I(S) \Rightarrow I \subseteq L(o)$, for some $o \in O$. Furthermore, if $I \subseteq L(o)$ for some $o \in O$, then $I = I \cap L(o)$, and hence $I \in I(S)$ follows from Eq. (4.18), which proves the claim.

Claim 2:

$$o_a = [o_b] \Leftrightarrow H_a(L_a(o_a)) = H_b(L_b(o_b)). \tag{4.28}$$

$$\begin{aligned}
& H_a(L_a(o_a)) \\
= & \left\{ \begin{array}{l} y \in Y : (\exists x_a \in X_a : H_a(x_a) = y) \\ \quad (\exists o_b \in O_b : o_a = [o_b]) \\ \quad (\exists x_b \in L_b(o_b) : (x_a, x_b) \in R) \end{array} \right\} \\
= & \left\{ \begin{array}{l} y \in Y : (\exists o_b \in O_b : o_a = [o_b]) \\ \quad (\exists x_b \in L_b(o_b) : H_b(x_b) = y) \end{array} \right\} \\
= & \; H_b(L_b(o_b)),
\end{aligned}$$

for any $o_b \in O_b$ such that $o_a = [o_b]$, proving the claim.

From Claim 1, it follows that:

$$\begin{aligned} H(I(S)) &= \left\{ \begin{array}{l} \overline{y} \in 2^Y : (\exists I \subseteq X)(\exists o \in O) \\ \quad \text{s.t. } \overline{y} = H(I) \wedge I \subseteq L(o) \end{array} \right\} \\ &= \{\overline{y} \in 2^Y : (\exists o \in O) \text{ s.t. } \overline{y} \subseteq H(L(o))\} \end{aligned} \tag{4.29}$$

Clearly, for every $o_b \in O_b$, there exists $o_a \in O_a$ such that $o_a = [o_b]$, and vice versa. It follows from this and Claim 2 that $\{\overline{y} \in 2^Y : (\exists o_a \in O_a) \text{ s.t. } \overline{y} = H_a(L_a(o_a))\} = \{\overline{y} \in 2^Y : (\exists o_b \in O_b) \text{ s.t. } \overline{y} = H_b(L_b(o_b))\}$. From Eq. (4.29), we therefore have that $H_a(I(S_a)) = H_b(I(S_b))$, which is Property 0. Property 3 follows immediately from Eq. (4.25), and Property 1 follows from this and the fact that R^{-1} is a function. Property 2 follows from Eq. (4.26) and the fact that $C(y)$ is a bijection for all $y \in Y$.

Claim 3: If $H_b(I_b) \subseteq H_a(I_a)$ and $o_a = [o_b]$, then $H_b(I_b \cap L_b(o_b)) \subseteq H_a(I_a \cap L_a(o_a))$.

Since H_b is a function, $H_b(I_b \cap L_b(o_b)) \subseteq H_b(I_b) \cap H_b(L_b(o_b))$. Since H_a is one-to-one, $H_a(I_a \cap L_a(o_a)) = H_a(I_a) \cap H_a(L_a(o_a))$. From Claim 2, $o_a = [o_b]$ implies $H_b(L_b(o_b)) = H_a(L_a(o_a))$. Thus, $H_b(I_b \cap L_b(o_b)) \subseteq H_b(I_b) \cap H_b(L_b(o_b)) \subseteq H_a(I_a) \cap H_a(L_a(o_a)) = H_a(I_a \cap L_a(o_a))$, proving the claim.

To prove Property 5, consider any $(I_b, u_b, I'_b) \in \Rightarrow_b$. From Properties 0 and 3, we have that there exists $I_a \in I(S_a)$ such that $(I_a, I_b) \in \overline{R}$. From Property 2, we have that there exists $u_a \in U_a$ such that $(u_a, u_b) \in \overline{C}(H_b(I_b))$. From the definition of $\Rightarrow_b$ in Eq. (4.19), we have that there exists $o_b \in O_b$ such that $I'_b = \cup_{x_b \in I_b}\{x'_b : (x_b, u_b, x'_b) \in \rightarrow_b\} \cap L_b(o_b)$. Now let $I^p_b = \cup_{x_b \in I_b}\{x'_b : (x_b, u_b, x'_b) \in \rightarrow_b\}$, $o_a = [o_b]$, $I'_a = \cup_{x_a \in I_a}\{x'_a : (x_a, u_a, x'_a) \in \rightarrow_a\} \cap L_a(o_a)$, and $I^p_a = \cup_{x_a \in I_a}\{x'_a : (x_a, u_a, x'_a) \in \rightarrow_a\}$, so that $I'_b = I^p_b \cap L_b(o_b)$ and $I'_a = I^p_a \cap L_a(o_a)$. From Property 5 of the state reduction, we have that, for all $(x_b, u_b, x'_b) \in \rightarrow_b$, there exists $(x_a, u_a, x'_a) \in \rightarrow_a$ such that $(x_a, x_b) \in R$ and $(x'_a, x'_b) \in R$. It therefore follows from Property 3 of the state reduction that $H_b(I^p_b) \subseteq H_a(I^p_a)$. From Claim 3, it then follows that $H_b(I'_b) = H_b(I^p_b \cap L_b(o_b)) \subseteq$

$H_a(I_a^p \cap L_a(o_a)) = H_a(I'_a)$. Now remark that, from the definition of I'_a and Eq. (4.19), we have that $(I_a, u_a, I'_a) \in \Rightarrow_a$. From Property 0, we have that there exists $I''_a \in I(S_a)$ such that $H_a(I''_a) = H_b(I'_b) \subseteq H_a(I'_a)$. Property 3 then gives $(I''_a, I'_b) \in \overline{R}$ and the fact that H_a is one-to-one gives $I''_a \subseteq I'_a$, completing the proof. □

Note, however, that Proc. IV.27 is not guaranteed to produce a state estimate reduction when S_a is a state reduction of S_b, since Property 4 may fail to hold between $\overline{S}_a$ and $\overline{S}_b$. Moreover, if S_a is an exact state reduction of S_b, then all of properties 4-7 of the exact state estimate reduction may fail to hold between $\overline{S}_a$ and $\overline{S}_b$. See Ex. IV.29 for an example where Property 4 of the state estimate reduction and Property 5 of the exact state estimate reduction fail to hold.

Example IV.29. Consider the system S_b of Fig. 4.3. It can be verified that system S_a of Fig. 4.4 is an exact state reduction of S_b. Now define S'_b as the partially observed system S_b, with the set of observations $O_b = \{o_{b,1}, o_{b,2}, o_{b,3}\}$ and observation function $L_b : O_b \to 2^{X_b}$ given by $L_b(o_{b,1}) = \{1\}$, $L_b(o_{b,2}) = \{2, 3\}$, and $L_b(o_{b,3}) = \{4, 5, 6\}$. Following Proc. IV.27, we find that $H_b(L_b(o_{b,1})) = H_b(L_b(o_{b,2})) = \{y_1\}$ and $H_b(L_b(o_{b,3})) = \{y_2, y_3\}$ and hence we define $O_a = \{o_{a,1}, o_{a,2}\}$ with $[o_{b,1}] = [o_{b,2}] = o_{a,1}$, $[o_{b,3}] = o_{a,2}$, $L_a(o_{a,1}) = \{y_1\}$, and $L_a(o_{a,2}) = \{y_2, y_3\}$. From Eqs. (4.18) and (4.19), we obtain $\Rightarrow_b = \{(\{1\}, u, \{4\}), (\{2\}, u, \{5\}), (\{3\}, u, \{6\}), (\{2, 3\}, u, \{5, 6\})\}$. On the other hand, $\Rightarrow_a = \{(\{y_1\}, u, \{y_2, y_3\})\}$. This violates Property 4 of the state estimate reduction, since there does not exist $(I_b, u_b, I'_b) \in \Rightarrow_b$ such that $H_b(I'_b) = \{y_2, y_3\}$. Furthermore, it also violates Property 5 of the *exact* state estimate reduction, since there doesn't exist $(I_a, u_a, I'_a) \in \Rightarrow_a$ satisfying either $H_a(I'_a) = \{y_2\}$ or $H_a(I'_a) = \{y_3\}$. In fact, there does not exist *any* set of observations O_a and observation function $L_a : O_a \to 2^{X_a}$ for partially observed system S'_a such that the estimator system of S'_a will be a state estimate reduction of $\overline{S}_b$. Because $H_b(L_b(o_{b,3})) = \{y_2, y_3\}$, satisfaction of Property 0 requires that there exist some $o_a \in O_a$ with $H_a(L_a(o_a)) \supseteq \{y_2, y_3\}$. But $H_b(L_b(o_{b,1})) = \{y_1\}$, so that satisfaction of Property 0 also requires

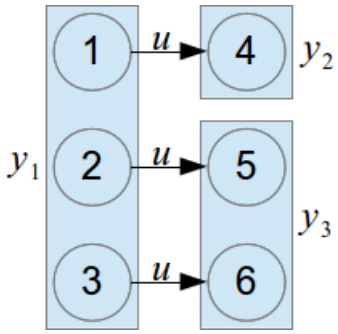


Figure 4.3: An example system S_b. Rectangles are used to denote states with the same output.

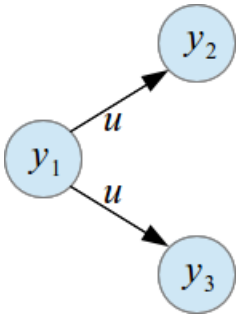


Figure 4.4: The system S_a that is an exact state reduction of system S_b of Fig. 4.3.

that there exist some $I_a \in I(S_a)$ such that $y_1 \in H_a(I_a)$. For this I_a and o_a, we have $\cup_{x_a \in I_a}\{x'_a \in X_a : (x_a, u, x'_a) \in \rightarrow_a\} \cap L_a(o_a) = \{y_2, y_3\}$, yielding $(I_a, u, \{y_2, y_3\}) \in \Rightarrow_a$ and hence the same violation of Property 4 as before. As previously noted, it is still possible to construct a state estimate reduction of $\overline{S}_b$ in such cases, but this must be accomplished by directly abstracting $\overline{S}_b$, rather than by abstracting S_b and constructing the estimator of this abstraction. □

4.6 Conditions for State Estimate Reductions and Exact State Estimate Reductions

We next turn attention to the vehicle control problem of Prob. IV.1. Denote by S_b the time-discretized system of Eq. (4.2) and by S'_b the partially observed system S_b with observations given by Eq. (4.4). We show in this section that, when the

discretization parameters μ and τ are chosen so that $e_{\max} = k\mu\tau/2$ for some $k \in \mathbb{N}$, then $Obs(G')$ is a state estimate reduction of the estimator $\overline{S}_b$ of S'_b. Furthermore, when G is an exact state reduction of S_b, we show that $Obs(G')$ is an exact state estimate reduction of $\overline{S}_b$. To prove these results, we would like to treat G as a transition system as in Def. IV.3, G' as a partially observed transition system as in Def. IV.16, and $Obs(G')$ as an estimator system as in Def. IV.17. This will require defining a translation between the various types of transition systems used in this work and the automata models used in supervisory control theory of DES, as well as some unification of notation. This translation is described in Sec. 4.6.1, parts of which are recalled from *Dallal et al.* (2014).

4.6.1 Translating Between Transition Systems and Discrete Event Systems

In Thm. IV.10, we obtain the maximally permissive, safe, and non-deadlocking supervisor σ_b of a system $S_b = (X_b, U_b, \rightarrow_b, Y_b, H_b)$ with respect to a safety specification $Safe_b \subseteq \rightarrow_b$ and set of marked states $X_{m,b} \subseteq X_b$, which is done by first constructing the maximally permissive, safe, and non-deadlocking supervisor σ_a of the abstracted system $S_a = (X_a, U_a, \rightarrow_a, Y_a, H_a)$ with respect to the induced safety and marking specifications $Safe_a \subseteq \rightarrow_a$ and $X_{m,a} \subseteq X_a$, and then using state and control relations to obtain σ_b from σ_a. In Thm. IV.22, we do the same, but with a system and its specifications defined over state estimates. In either of the above cases, the supervisor computation is achieved by constructing two DES automata G and H, where G (the system automaton) models the possible system behavior and H (the specification automaton) models the restricted system behavior that we wish to allow. In the fully observed case (or in the case of systems defined over state estimates), the supervisor is produced by solving the basic supervisory control problem in the non-blocking case (BSCP-NB). See *Cassandras and Lafortune* (2008); *Ramadge and*

Wonham (1987); *Wonham and Ramadge* (1987) for a description of this problem and associated algorithms for obtaining the supervisor.

We describe here how to construct equivalent automata for systems (as in Def. IV.3), partially observed systems (as in Def. IV.16), and estimator systems (as in Def. IV.17).

4.6.1.1 Translating Systems

Given system $S_a = (X_a, U_a, \rightarrow_a, Y_a, H_a)$ along with safety and marking specifications $Safe_a$ and $X_{m,a}$, the equivalent DES system automaton is $G := (X_a \cup Z_a, E_c \cup E_{uc}, \psi_G, x_{a,0}, X_{m,a})$, where:

$$E_c = U_a \tag{4.30}$$

$$\psi_G \subseteq (X_a \times E_c \times Z_a) \cup (Z_a \times E_{uc} \times (X_a \cup Z_a)) \tag{4.31}$$

$$\psi_G(x_a, u_a)! \Leftrightarrow u_a \in U_a(x_a) \tag{4.32}$$

$$\exists t \in E_{uc}^* : \psi_G(x_a, u_a t) = x'_a \Leftrightarrow (x_a, u_a, x'_a) \in \rightarrow_a . \tag{4.33}$$

The sets E_c and E_{uc} are, respectively, the sets of controllable and uncontrollable events of G and the set Z_a is a set of intermediate states. All events are taken to be observable. Eq. (4.30) signifies that the set of controllable events of G is simply the set of control inputs of S_a. Eq. (4.31) signifies that controllable events are defined only from the states of X_a and lead only to states in Z_a, whereas uncontrollable events are defined only from states in Z_a and may lead either to states in X_a or to other states in Z_a. Eq. (4.32) signifies that controllable event u_a is defined from state x_a if and only if $\mathbf{Post}_{u_a}(x_a)$ is non-empty (recall the definition of $U_a(x_a)$ from Def. IV.3). Finally, Eq. (4.33) signifies that there exists a sequence of uncontrollable events following controllable event u_a from state x_a leading to state x'_a if and only if (x_a, u_a, x'_a) is a transition of S_a.

The initial state $x_{a,0}$ may be a single fixed initial state or a dummy initial state with transitions to each of some set of states $X_{a,0}$. In the absence of any initial state information, it is assumed that any state can be an initial state and hence that there are transitions from $x_{a,0}$ to each state in X_a.

Constructing the specification automaton H is achieved through the same process as constructing G, but with the system $(X_a, U_a, Safe_a, Y_a, H_a)$ instead. With H and G defined, problem BSCP-NB is solved, yielding the desired supervisor. Because H is a sub-automaton of G and controllable events are defined only from X_a, the supervisor will have the form $S : X_a \to 2^{U_a}$, rather than a more general language based form (see, e.g. (*Hadj-Alouane et al.*, 1994)).

In order to have the same notation when working with systems as in Def. IV.3 and DES as above, we use the notation $U(x) := \{u \in E_c : \psi(x, u)!\}$ and $\mathbf{Post}_u(x) := \{x' \in X_a : (\exists t \in E_{uc}^*)(x' = \psi(x, ut))\}$ for $x \in X_a$ and (in an abuse of notation) will write $(x, u, x') \in \psi$ if $x \in X_a$ and $x' \in \mathbf{Post}_u(x)$.

4.6.1.2 Translating Partially Observed Systems

Given partially observed system $S'_a = (X_a, U_a, \to_a, Y_a, H_a, O_a, L_a)$ along with safety and marking specifications $Safe_a$ and $X_{m,a}$, the equivalent DES system automaton is $G := (X_a \cup X_a^p \cup Z_a, E_c \cup E_{uc}, \psi_G, x_{a,0}, X_{m,a})$, where X_a^p is a copy of X_a

with $x_a \equiv x_a^p$ used to denote that $x_a^p \in X_a^p$ is the copy of $x_a \in X_a$ and:

$$E_c = U_a \tag{4.34}$$

$$E_{uc} = O_a \cup E_{uc,uo} \tag{4.35}$$

$$\psi_G \subseteq \begin{array}{c}(X_a^p \times O_a \times X_a) \cup (X_a \times E_c \times Z_a) \\ \cup (Z_a \times E_{uc,uo} \times (X_a^p \cup Z_a))\end{array} \tag{4.36}$$

$$\psi_G(x_a^p, o_a)! \Leftrightarrow \exists x_a \in L_a(o_a) : x_a \equiv x_a^p \tag{4.37}$$

$$\psi_G(x_a^p, o_a) = x_a \Leftrightarrow x_a \in L_a(o_a) : x_a \equiv x_a^p \tag{4.38}$$

$$\psi_G(x_a, u_a)! \Leftrightarrow u_a \in U_a(x_a) \tag{4.39}$$

$$\begin{array}{l}\exists t \in E_{uc,uo}^* : \psi_G(x_a, u_a t) = x_a^p \in X_a^p \\ \Leftrightarrow (x_a, u_a, x_a') \in \rightarrow_a : x_a' \equiv x_a^p.\end{array} \tag{4.40}$$

As before, the sets E_c and E_{uc} are, respectively, the sets of controllable and uncontrollable events of G and the set Z_a is a set of intermediate states. The observable events are $E_c \cup O_a$ and the unobservable events are $E_{uc,uo}$. Eq. (4.36) signifies that observation events in O_a are defined only from states in X_a^p and lead to states in X_a, controllable events in U_a are defined only from states in X_a and lead to states in Z_a, whereas the events of $E_{uc,uo}$ are defined only from states in Z_a and lead to states in either Z_a or X_a^p. Eqs. (4.37) and (4.38) signify that event $o_a \in O_a$ is defined from $x_a^p \in X_a^p$ if and only if x_a^p is the copy of some state $x_a \in X_a$ such that $x_a \in L(o_a)$, in which case the transition leads to x_a. This constitutes the correction step for the partially observed system. Finally, Eqs. (4.39) and (4.40) have the same interpretation as Eqs. (4.32) and (4.33), with the exception that the string $t \in E_{uc,uo}^*$ takes the system to a state in X_a^p rather than a state in X_a.

Initial states are dealt with in the same way as in the fully observed case, except that any transitions from a dummy initial state should lead to states in X_a^p, rather than to states in X_a.

4.6.1.3 Translating Estimator Systems

Given system defined over state estimates $\overline{S}_a = (I(S_a), U_a, \rightrightarrows_a, 2^{Y_a}, H_a)$ along with safety and marking specifications $\overline{Safe}_a$ and $I_{m,a}$, the equivalent DES system automaton is $\overline{G} := (I(S_a) \cup I^p(S_a), E_c \cup E_{uc}, \overline{\psi}_{\overline{G}}, I_{a,0}, I_{m,a})$, where:

$$E_c = U_a \tag{4.41}$$

$$\overline{\psi}_{\overline{G}} \subseteq \begin{array}{c} (I(S_a) \times E_c \times I^p(S_a)) \\ \cup (I^p(S_a) \times E_{uc} \times I(S_a)) \end{array} \tag{4.42}$$

$$\overline{\psi}_{\overline{G}}(I_a, u_a)! \Leftrightarrow \exists (I_a, u_a, I'_a) \in \rightrightarrows_a \tag{4.43}$$

$$\begin{array}{l} \exists e_{uc} \in E_{uc} : \overline{\psi}_{\overline{G}}(I_a, u_a e_{uc}) = I'_a \\ \Leftrightarrow (I_a, u_a, I'_a) \in \rightrightarrows_a \end{array} . \tag{4.44}$$

The sets E_c and E_{uc} are, respectively, the sets of controllable and uncontrollable events of $\overline{G}$ and the set $I^p(S_a)$ is a set of intermediate states. All events are taken to be observable. Eqs. (4.41)-(4.44) have the same interpretations as Eqs. (4.30)-(4.33) in the fully observed case, except for the fact that events in e_{uc} lead directly to states in $I(S_a)$, rather than leading to states in either $I(S_a)$ or $I^p(S_a)$.

It can be shown that, if $\overline{G}$ is obtained as $Obs(G)$ for some DES abstraction G representing a partially observed system, then $\overline{G}$ will indeed have the above form. In this case, we will additionally have the following properties:

$$E_{uc} = O_a \tag{4.45}$$

$$(I_a, u_a, I^p_a) \in \overline{\psi}_{\overline{G}} \Leftrightarrow I^p_a \cap X^p_a \equiv I^p_a(I_a, u_a) \tag{4.46}$$

$$(I^p_a, o_a, I'_a) \in \overline{\psi}_{\overline{G}} \Leftrightarrow I'_a = I^c_a(I^p_a \cap X^p_a, o_a), \tag{4.47}$$

where $I^p_a(\cdot,\cdot)$ and $I^c_a(\cdot,\cdot)$ are the prediction and correction functions of Eqs. (4.16) and (4.17). In words, the states of $I^p(S_a)$ represent state estimates reached after prediction steps, whereas states of $I(S_a)$ represent state estimates reached after cor-

rection steps. However, because the observer is constructed through the unobservable reach operation, the states of $I^p(S_a)$ will be subsets of $Z_a \cup X_a^p$, which is why we use $I_a^p \cap X_a^p$ rather than just I_a^p in Eqs. (4.46) and (4.47). *Thus, constructing the observer of a DES automaton representing a partially observed system yields a DES automaton representing the estimator of the partially observed system.*

The initial state $I_{a,0}$ may be a single fixed initial state or a dummy initial state. In the latter case, there are observable transitions from the dummy initial state to each state of some subset of $I(S_a)$. In the absence of any initial state information, it is assumed that any state can be an initial state and hence that there are transitions from $I_{a,0}$ to each state in $I(S_a)$. Constructing the specification automaton $\overline{H}$ and solving for the desired supervisor is *not* done by taking the observer of any system. Instead, it must be constructed from $\overline{G}$, by pruning the transitions of $\overline{\psi}$, in accordance with Defs. IV.20 and IV.21.

In order to have the same notation when working with systems as in Def. IV.17 and DES as above, we will abuse notation and write $(I, u, I') \in \overline{\psi}$ if $I_a \in I(S_a)$ and there exists some $e_{uc} \in E_{uc}$ such that $I' = \overline{\psi}(I_a, ue_{uc})$.

4.6.2 Proofs of state estimate reductions between the observer and the continuous estimator

We now prove the results relating state reductions to state estimate reductions and exact state reductions to exact state estimate reductions, as discussed at the beginning of this section.

Remark IV.30. In the theorems that follows, we will limit attention to state estimates that are boxes (i.e., products of intervals). This is justified by the fact that both the prediction and correction functions of Eqs. (4.5) and (4.6) map boxes to boxes. Thus, if the initial state estimate is a box, then all future state estimates will be boxes as well. Furthermore, because both dynamics and estimation are uncoupled

(estimation is uncoupled in the sense that a measurement χ_i of vehicle i's position tells us nothing about vehicle $j \neq i$'s position), we can establish the existence of some transition (I, u, I') by examining each vehicle in turn.

Before proceeding, we define notation for the following proofs. For any two vectors $a, b \in \mathbb{R}^n$, define $[a, b] := \{x \in \mathbb{R}^n : a_i \leq x_i \leq b_i, i = 1, \ldots, n\}$. For any $q_{i,lo}, q_{i,hi} \in \tilde{Q}_i$, we will use $\{[q_{i,lo}, q_{i,hi}]\}$ to denote the set $\{q_{i,lo}, q_{i,lo} + \mu\tau, \ldots, q_{i,hi}\}$. Next, let $w_{\min} = \mu\tau\lfloor\delta_{\min}/(\mu\tau)\rfloor$, $w_{\max} = \mu\tau\lceil\delta_{\max}/(\mu\tau)\rceil$, $\epsilon_{\min} = \delta_{\min} - w_{\min}$, and $\epsilon_{\max} = w_{\max} - \delta_{\max}$, so that $0 \leq \epsilon_{\min}, \epsilon_{\max} < \mu\tau$. Additionally, note that we will generally use I to denote state estimates for the continuous system and ι to denote state estimates for the abstracted system. The notation I_b and I_a will continue to be used only for general results (i.e., not specific to the vehicle control problem under consideration). Finally, take S_b to be the system of Eq. (4.2) with the output map of Eq. (4.3), S'_b to be the partially observed system S_b with observations given by Eq. (4.4), S_a to be the state reduction G of S_b, and S'_a to be G'. Then, $X_b = X$, $X_a = Y_a = Y_b = \tilde{Q}$, $U_b = V_c$, $U_a = U_c$, $H_b = \ell$, H_a is the identity map, $O_b = X$, $L_b = L$, $O_a = \Lambda$, $L_a = L^\Lambda$, $I^p_b = I^p$, $I^c_b = I^c$, $I^p_a = \overline{\psi}^p$, and $I^c_a = \overline{\psi}^c$.

Proposition IV.31. *Let $\overline{S}_b$ be the estimator system of the system of Eq. (4.2) with the output map of Eq. (4.3) and observations of Eq. (4.4). If $e_{\max} = k\mu\tau/2$ for some $k \in \mathbb{N}$, then $Obs(G')$ is a state estimate reduction of $\overline{S}_b$, with state and control relations $\overline{R}$ and $\overline{C}$ given by Eqs. (4.25) and (4.26).*

The proof is contained in the appendix.

Theorem IV.32. *Suppose that $e_{\max} = k\mu\tau/2$ for some $k \in \mathbb{N}$, let $\overline{G} = Obs(G')$, as described in Sec. 4.5.1, and let $\overline{H}$ be the specification automation obtained by pruning the transitions of $\overline{G}$, in accordance with Defs. IV.20 and IV.21. Solve for the supremal controllable sublanguage $(\mathcal{L}_m(\overline{H}))^{\uparrow C}$ of $\mathcal{L}_m(\overline{H})$ with respect to $\mathcal{L}(\overline{G})$ and uncontrollable event set $E_{uc} = \Lambda$, obtaining a maximally permissive safe and non-*

blocking supervisor $S : 2^{\tilde{Q}} \to 2^{U_c}$. *Then the supervisor* $\sigma : 2^X/\ell \to 2^{V_c}$ *defined by* $v_c \in \sigma(I) \Leftrightarrow u_c = \tau v_c \in S(\ell(I))$ *solves Prob. IV.1.*

Proof. Immediate from Thm. IV.22 and Prop. IV.31. □

Next, we prove the equivalent theorem for the case where G is an exact state reduction of the system of Eq. (4.2) with the output map of Eq. (4.3). We first prove the following lemma, however.

Lemma IV.33. *For any two intervals* $I^1 = [I^1_{lo}, I^1_{hi}]$ *and* $I^2 = [I^2_{lo}, I^2_{hi}]$ *such that* $I^1 \cap I^2 \neq \emptyset$, $\ell_i(I^1 \cap I^2) = \ell_i(I^1) \cap \ell_i(I^2)$.

Proof. Given any non-empty interval $I = [I_{lo}, I_{hi}]$, we have $\ell_i(I) = \{[\ell_i(I_{lo}), \ell_i(I_{hi})]\}$. Since ℓ_i is monotonic, we have $\min\{\ell_i(x_1), \ell_i(x_2)\} = \ell_i(\min\{x_1, x_2\})$ and $\max\{\ell_i(x_1), \ell_i(x_2)\} = \ell_i(\max\{x_1, x_2\})$, for any $x_1, x_2 \in \mathbb{R}$. It follows that $I^1 \cap I^2 = [\max\{I^1_{lo}, I^2_{lo}\}, \min\{I^1_{hi}, I^2_{hi}\}]$ and, if $I^1 \cap I^2 \neq \emptyset$, then $\ell_i(I^1 \cap I^2) = \{[\ell_i(\max\{I^1_{lo}, I^2_{lo}\}), \ell_i(\min\{I^1_{hi}, I^2_{hi}\})]\}$
$= \{[\max\{\ell_i(I^1_{lo}), \ell_i(I^2_{lo})\}, \min\{\ell_i(I^1_{hi}), \ell_i(I^2_{hi})\}]\} = \{[\ell_i(I^1_{lo}), \ell_i(I^1_{hi})]\} \cap \{[\ell_i(I^2_{lo}), \ell_i(I^2_{hi})]\} =$
$\ell_i(I^1) \cap \ell_i(I^2)$. □

Proposition IV.34. *Let* $\overline{S}_b$ *be the estimator system of the system of Eq. (4.2) with the output map of Eq. (4.3) and observations of Eq. (4.4). If* δ_{min} *and* δ_{max} *are both integer multiples of* $\tau\mu$ *and* $e_{\max} = k\mu\tau/2$ *for some* $k \in \mathbb{N}$, *then* $Obs(G')$ *is an exact state estimate reduction of* $\overline{S}_b$, *with state and control relations* $\overline{R}$ *and* $\overline{C}$ *given by Eqs. (4.25) and (4.26), and set of reachable state estimates* $I'(S_b)$ *as follows:*

$$\begin{aligned} I'(S_b) &= \prod_{i \in \mathcal{N}} I'_i(S_b) \\ I'_i(S_b) &= \left\{ \begin{array}{l} [q_{i,lo} + \epsilon_i, q_{i,hi} + \epsilon_i] \ s.t. \\ \quad q_{i,lo}, q_{i,hi} \in \tilde{Q}_i \wedge -\mu\tau/2 < \epsilon_i \leq \mu\tau/2 \end{array} \right\} \end{aligned} \tag{4.48}$$

Proof. Beyond what has been shown in Prop. IV.31, it remains to prove that Properties 5-7 of Def. IV.24 also hold. We begin with Property 5. Consider any transition

$(I, v_c, I') \in \Rightarrow_b$ and let χ be such that $I' = I^c(I^p(I, v_c), \chi)$. We show that $\iota' = \overline{\psi}^c(\overline{\psi}^p(\iota, u_c), \lambda)$, for $\iota = \ell(I)$, $u_c = v_c\tau$, $\iota' = \ell(I')$, and $\lambda = [\chi]$, from which it follows that $(\iota, u_c, \iota') \in \overline{\psi}$, as required for Property 5. As in the proof of Prop. IV.31, we consider one vehicle at a time. Let $I_i = [x_{i,lo}, x_{i,hi}]$ and $\iota_i = \ell_i(I_i) = \{[q_{i,lo}, q_{i,hi}]\}$. Since δ_{min} and δ_{max} are both integer multiples of $\tau\mu$, we have $I_i^p(I, v_c) = [x_{i,lo}+k_{i,lo}\mu\tau, x_{i,hi}+k_{i,hi}\mu\tau]$ and $\overline{\psi}^p(\iota, u_c) = \{[q_{i,lo} + k_{i,lo}\mu\tau, q_{i,hi} + k_{i,hi}\mu\tau]\}$ for some $k_{i,lo}, k_{i,hi} \in \mathbb{N}$. Now, for any $x_i \in \mathbb{R}$ and $c_i \in \mathbb{Z}$, $\ell_i(x_i + c_i\mu\tau) = \ell_i(x_i) + c_i\mu\tau$. Since $\iota_i = \ell_i(I_i)$, it therefore follows that $\overline{\psi}_i^p(\iota, u_c) = \ell_i(I_i^p(I, v_c))$. Finally, $L_i^\Lambda(\lambda) = \ell_i(L_i(\chi))$ and $\iota_i' = \overline{\psi}_i^c(\overline{\psi}^p(\iota, u_c), \lambda) = \overline{\psi}_i^p(\iota, u_c) \cap L_i^\Lambda(\lambda) = \ell_i(I_i^p(I, v_c)) \cap \ell_i(L_i(\chi)) = \ell_i(I_i^p(I, v_c) \cap L_i(\chi)) = \ell_i(I')$, using Lemma IV.33.

We next prove Property 6. Consider any transition $(\iota, u_c, \iota') \in \overline{\psi}$ and any box $I' = [I'_{lo}, I'_{hi}] \in I'(S_b)$ such that $\ell(I') = \iota'$ and let $\lambda \in \Lambda$ be such that $\iota' = \overline{\psi}^c(\overline{\psi}^p(\iota, u_c), \lambda)$. We construct $I \in I'(S_b)$, $v_c = u_c/\tau$, and χ such that $\ell(I) = \iota$, and $I' = I^c(I^p(I, v_c), \chi)$, from which it follows that $(I, v_c, I') \in \Rightarrow_b$, as required for Property 6. As before, we consider a single vehicle at a time. Let $\iota_i = \{[q_{i,lo}, q_{i,hi}]\}$, $\iota_i^p = \overline{\psi}^p(\iota, u_c) = \{[q_{i,lo}^p, q_{i,hi}^p]\}$, and $L_i^\Lambda(\lambda) = \{[q_{i,lo}^o, q_{i,hi}^o]\}$, so that $\iota_i' = \overline{\psi}_i^c(\overline{\psi}^p(\iota, u_c), \lambda) = \overline{\psi}_i^p(\iota, u_c) \cap L_i^\Lambda(\lambda) = \{[\max\{q_{i,lo}^p, q_{i,lo}^o\}, \min\{q_{i,hi}^p, q_{i,hi}^o\}]\}$. Given Eq. (4.48), and the fact that $\ell(I') = \iota'$, it must be that $I_i' = [q_{i,lo}' + \epsilon, q_{i,hi}' + \epsilon]$ for some $\epsilon \in (-\mu\tau/2, \mu\tau/2]$. Take $I_i = [q_{i,lo} + \epsilon, q_{i,hi} + \epsilon]$ and $\chi_i = (q_{i,lo}^o + q_{i,hi}^o)/2 + \epsilon$, so that $\ell_i(I_i) = \iota_i$, $I_i^p(I, v_c) = [q_{i,lo}^p + \epsilon, q_{i,hi}^p + \epsilon]$, $\ell_i(I_i^p(I, v_c)) = \iota_i^p$, and $L_i(\chi) = [q_{i,lo}^o + \epsilon, q_{i,hi}^o + \epsilon]$. Thus, $I_i^c(I_i^p(I, v_c), \chi) = [q_{i,lo}^p + \epsilon, q_{i,hi}^p + \epsilon] \cap [q_{i,lo}^o + \epsilon, q_{i,hi}^o + \epsilon] = [\max\{q_{i,lo}^p, q_{i,lo}^o\} + \epsilon, \min\{q_{i,hi}^p, q_{i,hi}^o\} + \epsilon]$, from which it follows that $\ell_i(I_i^c(I_i^p(I, v_c), \chi)) = \iota_i'$. This implies that $I_i' = I_i^c(I^p(I, v_c), \chi)$ (otherwise it could not be that $\ell(I_i') = \iota_i'$), which proves Property 6.

Finally, we prove Property 7. By assumption, all state estimates are boxes, so that any $\overline{y} \in \overline{Y}$ has the form $\overline{y} = \prod_{i \in \mathcal{N}} \{[q_{i,lo}, q_{i,hi}]\}$. Once again, we consider one vehicle at a time. Since $\overline{y}_i = \{[q_{i,lo}, q_{i,hi}]\}$, $\ell_i^{-1}(\overline{y}_i) = (q_{i,lo} - \mu\tau/2, q_{i,hi} + \mu\tau/2] =$

$\bigcup_{\epsilon\in(-\mu\tau/2,\mu\tau/2]}[q_{i,lo}+\epsilon, q_{i,hi}+\epsilon]$. Since each of the intervals $[q_{i,lo}+\epsilon, q_{i,hi}+\epsilon]$ are in the set $I_i'(S_b)$, this proves Property 7. □

Theorem IV.35. *If $e_{\max} = k\mu\tau/2$ for some $k \in \mathbb{N}$ and δ_{min} and δ_{max} are both integer multiples of $\tau\mu$, then the supervisor σ of Thm. IV.32 solves Prob. IV.1, and is maximally permissive among the class of all supervisors, not merely memoryless ones.*

Proof. Immediate from Thm. IV.26 and Prop. IV.34. □

The solution method presented in this work is depicted in Fig. 4.5. It should be noted that for a DES automaton with m states, the observer of the DES automaton can have up to 2^m states. However, because state estimates in the system under consideration will always be boxes, the number of states of $\overline{G} = Obs(G')$ is quadratic in $|\tilde{Q}|$ (any box is uniquely parametrized by two states). Furthermore, because the specification automaton $\overline{H}$ is a subautomaton of $\overline{G}$ and $\overline{G}$ is acyclic (by virtue of the asumption that velocities are bounded below by μ), the computation of the supervisor in Thm. IV.32 takes only time linear in the size of $\overline{G}$ (see, e.g. (*Hadj-Alouane et al.*, 1994)). See *Dallal et al.* (2014) for a more thorough discussion of running time.

4.7 Conclusion

We considered the problem of finding a safe, non-deadlocking, and maximally permissive supervisor for a set of vehicles at an intersection, in the presence of uncontrolled vehicles, bounded disturbances, and measurement uncertainty. We showed how to construct a suitable DES abstraction for this partially observed system, by discretizing the system in space and time, grouping the set of possible measurements into a finite set of equivalence classes, and using a combination of uncontrollable and unobservable events to model the uncontrolled vehicles, the disturbance, and the measurement uncertainty. We described a procedure for obtaining supervisors

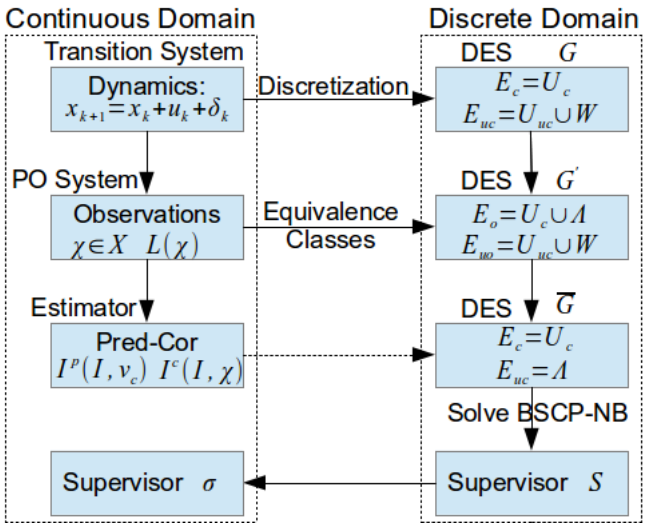


Figure 4.5: The solution method. Given the time discretized system of Eq. (4.2) with the space discretization given by $\ell(\cdot)$ of Eq. (4.3), constituting system S_b, we can construct DES abstraction G that is a state reduction of this system. Given system S_b with measurement uncertainty given by $L(\cdot)$ of Eq. (4.4), constituting partially observed system S_b', we can construct DES abstraction G' that models the measurement uncertainty of $L(\cdot)$ by partitioning the set of measurements X into equivalence classes Λ, in accordance with Proc. IV.27. Given the estimator $\overline{S}_b$ of partially observed system S_b', we can construct DES abstraction $\overline{G}$ that is a state estimate reduction of $\overline{S}_b$. Furthermore, when $e_{\max} = k\mu\tau/2$ for some $k \in \mathbb{N}$, $\overline{G}$ can be obtained as the observer of G' (Props. IV.31 and IV.34). A DES supervisor S is then computed by solving problem BSCP-NB, from which a continuous domain supervisor σ is obtained which solves Prob. IV.1 (Thms. IV.32 and IV.35).

for partially observed systems, by abstracting the system and specifications, solving for a supervisor in the abstracted domain, and translating this supervisor back to the original problem domain. We presented the general notions of state estimate reduction and exact state estimate reduction relating prediction-correction estimator systems to abstractions and showed that the obtained supervisors will be maximally permissive among the class of memoryless supervisors, in the case of a state estimate reduction, or maximally permissive among the class of all supervisors, in the case of an exact state estimate reduction. Finally, we showed that, when the discretization parameters are properly chosen for the vehicle control problem under consideration,

it is possible to obtain a state estimate reduction or an exact state estimate reduction. Future work includes generalizing the methods presented here to systems with more complex dynamics, and extensions to stochastic problem formulations.

Appendix: Proof of Prop. IV.31

Proof. First, note that Prop. IV.11 shows that G is a state reduction of the system of Eq. (4.2) with the output map of Eq. (4.3). Furthermore, Prop. IV.11 defines the control relation C of this state reduction by $C(q) := \{(u_c, v_c) : v_c\tau = u_c \in U_c\}$, for all $q \in \tilde{Q}$ and therefore C satisfies the property that $q_1, q_2 \in \overline{y} \Rightarrow C(q_1) = C(q_2)$, for all $\overline{y} \in H_a(I(S'_a)) \cup H_b(I(S'_b))$. Finally, G' is the DES automaton equivalent of a partially observed system constructed in accordance with Proc. IV.27 and hence $Obs(G')$ is the DES automaton equivalent of the corresponding estimator system, as per the discussion of Sec. 4.6.1.3. Therefore, the conditions of Thm. IV.28 are satisfied and it remains only to prove that Property 4 of the state estimate reduction also holds. We begin by proving the following claim:

Claim 4: Consider any two systems S_a and S_b such that S_a is a state reduction of S_b. For any $I_a \subseteq X_a$, $u_a \in U_a$, and $u_b \in U_b$: $(u_a, u_b) \in \overline{C}(H_a(I_a))$, let $I_b = \{x_b \in X_b : \exists x_a \in I_a \text{ s.t. } (x_a, x_b) \in R\}$. Then $(I^p_a(I_a, u_a), I^p_b(I_b, u_b)) \in \overline{R}$.

Recall from Eq. (4.16) that $I^p(I, u) = \cup_{x\in I}\{x' \in X : \exists(x, u, x') \in \rightarrow\}$. Consider any $(x_b, u_b, x'_b) \in \rightarrow_b$ with $x_b \in I_b$ and $x'_b \in I^p_b(I_b, u_b)$. By Property 1 of Def. IV.8, there is a unique $x_a \in X_a$ such that $(x_a, x_b) \in R$ and a unique x'_a such that $(x'_a, x'_b) \in R$. From the definitions of I_a and I_b above, $x_a \in I_a$. From Property 5 of Def. IV.8, it must be that $(x_a, u_a, x'_a) \in \rightarrow_a$, so that $x'_a \in I^p_a(I_a, u_a)$. Conversely, consider any $(x_a, u_a, x'_a) \in \rightarrow_a$ with $x_a \in I_a$ and $x'_a \in I^p_a(I_a, u_a)$. By Property 4 of Def. IV.8, there must exist some $(x_b, u_b, x'_b) \in \rightarrow_b$ with $(x_a, x_b) \in R$ and $(x'_a, x'_b) \in R$. From the definitions of I_a and I_b above, it follows that any such x_b is in I_b, and hence that $x'_b \in I^p_b(I_b, u_b)$, which completes the proof of the claim.

The remaining property to be proven states that, for all $(\iota, u_c, \iota') \in \overline{\psi}$, there exists $(I, v_c, I') \in \rightrightarrows_b$ such that $(\iota, I) \in \overline{R}$, $(u_c, v_c) \in \overline{C}(\iota)$, and $(\iota', I') \in \overline{R}$. As stated in Remark IV.30, it is sufficient to consider each vehicle separately. Thus, we wish to construct, for each vehicle i, an interval $I_i = [x_{i,lo}, x_{i,hi}]$ such that: $x_{i,hi} - x_{i,lo} \leq 2e_{\max}$; $\ell_i(I_i) = \iota_i$; and $\ell_i(I_i') = \iota_i'$, for $I_i' = I_i^c(I_i^p(I_i))$.

Suppose that $e_{\max} = k\tau\mu/2$. Then, for any $\chi \in X$, $L_i(\chi) = [\chi_i - k\tau\mu/2, \chi_i + k\tau/mu/2]$, which is an interval of size $k\tau\mu$. It follows that $\ell_i(L_i(\chi))$ has the form $\{[q_i, q_i + k\mu\tau]\}$, for some $q_i \in \tilde{Q}_i$. From Claims 1 and 2 of the proof of Thm. IV.28, it follows that any $\iota \in I(G')$ has the form $\iota = \prod_{i=1}^n \{[q_{i,lo}, q_{i,hi}]\}$, where $q_{i,lo}, q_{i,hi} \in \tilde{Q}_i$ and $0 \leq q_{i,hi} - q_{i,lo} \leq k\tau\mu$, for all $i = 1, \ldots, n$ (i.e., the discrete version of a box). Consider some transition $(\iota, u_c, \iota') \in \overline{\psi}$. For vehicle i, we have $\iota_i = \{[q_{i,lo}, q_{i,hi}]\}$ and $\iota_i' = \{[q_{i,lo}', q_{i,hi}']\}$. Let $\iota_i^p = \overline{\psi}^p(\iota, u_c) := \{[q_{i,lo}^p, q_{i,hi}^p]\}$. For a controlled vehicle, this is given by $\iota_i^p = \{[q_{i,lo} + u_{c,i} + w_{\min}, q_{i,hi} + u_{c,i} + w_{\max}]\}$ whereas for an uncontrolled vehicle this is given by $\iota_i^p = \{[q_{i,lo} + u_{\min} + w_{\min}, q_{i,hi} + u_{\max} + w_{\max}]\}$. For any $\lambda \in \Lambda$, we have that $L_i^\Lambda(\lambda) = \{[q_{i,lo}^o, q_{i,hi}^o]\}$, for some $q_{i,lo}^o, q_{i,hi}^o \in \tilde{Q}_i$ such that $q_{i,hi}^o - q_{i,lo}^o = k\mu\tau$. It follows that $\iota_i' = \iota_i^p \cap L_i^\Lambda(\lambda) = \{[\max\{q_{i,lo}^o, q_{i,lo}^p\}, \min\{q_{i,hi}^o, q_{i,hi}^p\}]\}$. Now remark that $\ell_i^{-1}(\iota_i) = (q_{i,lo} - \mu\tau/2, q_{i,hi} + \mu\tau/2]$ and that from Claim 4 we therefore have $(\iota^p, I^p((q_{i,lo} - \mu\tau/2, q_{i,hi} + \mu\tau/2], v_c)) \in \overline{R}$. Since $\ell_i^{-1}(q_i) = (q_i - \mu\tau/2, q_i + \mu\tau/2]$, it follows that $I_i^p((q_{i,lo} - \mu\tau/2, q_{i,hi} + \mu\tau/2], v_c) = (\beta_{i,lo}, \beta_{i,hi}]$ for some $\beta_{i,lo}$ and $\beta_{i,hi}$ such that $q_{i,lo}^p - \mu\tau/2 < \beta_{i,lo} \leq q_{i,lo}^p + \mu\tau/2$ and $q_{i,hi}^p - \mu\tau/2 < \beta_{i,hi} \leq q_{i,hi}^p + \mu\tau/2$. This fact will be used in the analysis of each of four cases separately:

Case 1: $q_{i,lo}^o > q_{i,lo}^p$ and $q_{i,hi}^o < q_{i,hi}^p$.

This gives $\iota_i' = \{[q_{i,lo}^o, q_{i,hi}^o]\}$. Take $I_i = [q_{i,lo}, q_{i,hi}]$ and $\chi_i = (q_{i,lo}^o + q_{i,hi}^o)/2$ so that $I_i^p(I, v_c) = [\beta_{i,lo} + \mu\tau/2, \beta_{i,hi} - \mu\tau/2] := [x_{i,lo}^p, x_{i,hi}^p]$, $\ell_i(I_i) = \{[q_{i,lo}, q_{i,hi}]\} = \iota_i$ and $L_i(\chi) = [q_{i,lo}^o, q_{i,hi}^o]$. Since $q_{i,lo}^o > q_{i,lo}^p$ and $q_{i,hi}^o < q_{i,hi}^p$, it must be that $q_{i,lo}^o \geq q_{i,lo}^p + \mu\tau$ and $q_{i,hi}^o \leq q_{i,hi}^p - \mu\tau$. Thus, $x_{i,lo}^p = \beta_{i,lo} + \mu\tau/2 \leq q_{i,lo}^p + \mu\tau \leq q_{i,lo}^o$ and $x_{i,hi}^p = \beta_{i,hi} - \mu\tau/2 > q_{i,hi}^p - \mu\tau \geq q_{i,hi}^o$, from which it follows that $I_i' = I_i^c(I^p(I, v_c), \chi) =$

$[x^p_{i,lo}, x^p_{i,hi}] \cap [q^o_{i,lo}, q^o_{i,hi}] = [q^o_{i,lo}, q^o_{i,hi}]$, so that $\ell_i(I'_i) = \{[q^o_{i,lo}, q^o_{i,hi}]\} = \iota'_i$.

Case 2: $q^o_{i,lo} > q^p_{i,lo}$ and $q^o_{i,hi} \geq q^p_{i,hi}$.

This gives $\iota'_i = \{[q^o_{i,lo}, q^p_{i,hi}]\}$. Consider some ϵ such that $0 \leq \epsilon_{\max} < \epsilon < \mu\tau$ and take $I_i = [q_{i,lo} - \mu\tau/2 + \epsilon, q_{i,hi} - \mu\tau/2 + \epsilon]$ and $\chi_i = (q^o_{i,lo} + q^o_{i,hi})/2$, so that $I^p_i(I, v_c) = [\beta_{i,lo} + \epsilon, \beta_{i,hi} - \mu\tau + \epsilon] := [x^p_{i,lo}, x^p_{i,hi}]$ and $L_i(\chi) = [q^o_{i,lo}, q^o_{i,hi}]$. Since $0 < \epsilon < \mu\tau$, we have $\ell_i(I_i) = \{[q_{i,lo}, q_{i,hi}]\} = \iota_i$. Since $\epsilon_{\max} = w_{\max} - \delta_{\max} < \epsilon < \mu\tau$, we have $w_{\max} < \epsilon + \delta_{\max} < \mu\tau + \delta_{\max} \leq \mu\tau + w_{\max}$. It follows that $q_{i,hi} - \mu\tau/2 + w_{\max} < q_{i,hi} - \mu\tau/2 + \epsilon + \delta_{\max} < q_{i,hi} + \mu\tau/2 + w_{\max}$, and hence that $\ell_i(q_{i,hi} - \mu\tau/2 + \epsilon + \delta_{\max}) = q_{i,hi} + w_{\max}$. The same reasoning shows that we must have $\ell_i(x^p_{i,hi}) = q^p_{i,hi}$, and hence that $q^p_{i,hi} - \mu\tau/2 < x^p_{i,hi} \leq q^p_{i,hi} + \mu\tau/2$. Since $q^o_{i,lo} > q^p_{i,lo}$ and $q^o_{i,hi} \geq q^p_{i,hi}$, we have that $q^o_{i,lo} > q^p_{i,lo} + \mu\tau$ and $q^o_{i,hi} \geq q^p_{i,hi} + \mu\tau$. Using this and $0 < \epsilon < \mu\tau$ gives $x^p_{i,hi} = \beta_{i,hi} - \mu\tau + \epsilon < \beta_{i,hi} \leq q^p_{i,hi} + \mu\tau/2 \leq q^o_{i,hi} - \mu\tau/2 < q^o_{i,hi}$. Similarly, $x^p_{i,lo} = \beta_{i,lo} + \epsilon < \beta_{i,lo} + \mu\tau \leq q^p_{i,lo} + 3\mu\tau/2 < q^o_{i,lo} + \mu\tau/2$. Hence, $q^o_{i,lo} \leq \max\{x^p_{i,lo}, q^o_{i,lo}\} < q^o_{i,lo} + \mu\tau/2$. Also, it follows from $q^o_{i,hi} \geq q^p_{i,hi}$ and $q^p_{i,hi} - \mu\tau/2 < x^p_{i,hi} \leq q^p_{i,hi} + \mu\tau/2$ that $q^p_{i,hi} - \mu\tau/2 < \min\{x^p_{i,hi}, q^o_{i,hi}\} \leq q^p_{i,hi} + \mu\tau/2$. Finally, this gives $I'_i = I^c_i(I^p(I, v_c), \chi) = [x^p_{i,lo}, x^p_{i,hi}] \cap [q^o_{i,lo}, q^o_{i,hi}] = [\max\{x^p_{i,lo}, q^o_{i,lo}\}, \min\{x^p_{i,hi}, q^o_{i,hi}\}]$, so that $\ell_i(I'_i) = \{[q^o_{i,lo}, q^p_{i,hi}]\} = \iota'_i$.

Case 3: $q^o_{i,lo} \leq q^p_{i,lo}$ and $q^o_{i,hi} < q^p_{i,hi}$.

This gives $\iota'_i = \{[q^p_{i,lo}, q^o_{i,hi}]\}$. Consider some ϵ such that $0 \leq \epsilon_{\min} < \epsilon < \mu\tau$ and take $I_i = [q_{i,lo} + \mu\tau/2 - \epsilon, q_{i,hi} + \mu\tau/2 - \epsilon]$ and $\chi_i = (q^o_{i,lo} + q^o_{i,hi})/2$. The remainder is symmetrical to Case 2.

Case 4: $q^o_{i,lo} \leq q^p_{i,lo}$ and $q^o_{i,hi} \geq q^p_{i,hi}$.

This gives $\iota'_i = \{[q^p_{i,lo}, q^p_{i,hi}]\}$. Furthermore, since $q^o_{i,lo} \leq q^p_{i,lo}$ and $q^o_{i,hi} \geq q^p_{i,hi}$, we have that $q^p_{i,hi} - q^p_{i,lo} \leq q^o_{i,hi} - q^o_{i,lo} = k\mu\tau$. Also, it must always be that $q_{i,hi} - q_{i,lo} \leq q^p_{i,hi} - q^p_{i,lo}$, with equality possible only if i is a controlled vehicle and $\delta_{\min} = \delta_{\max} = 0$, so that $q_{i,hi} - q_{i,lo} \leq k\mu\tau$.

Suppose first that $q_{i,hi} - q_{i,lo} = k\mu\tau$. Then $k\mu\tau = q_{i,hi} - q_{i,lo} \leq q^p_{i,hi} - q^p_{i,lo} \leq q^o_{i,hi} - q^o_{i,lo} = k\mu\tau$, and hence it follows that vehicle i is controlled and $\delta_{\min} = \delta_{\max} = 0$. Thus,

$q^p_{i,lo} = q_{i,lo} + u_{c,i}$ and $q^p_{i,hi} = q_{i,hi} + u_{c,i}$. Therefore, taking $I_i = [q_{i,lo}, q_{i,hi}]$ and $\chi_i = (q^o_{i,lo} + q^o_{i,hi})/2$ yields $\ell_i(I_i) = \{[q_{i,lo}, q_{i,hi}]\} = \iota_i$, $L_i(\chi) = [q^o_{i,lo}, q^o_{i,hi}]$, $I^p_i(I, v_c) = [q^p_{i,lo}, q^p_{i,hi}]$, $I'_i = I^c_i(I^p(I, v_c), \chi) = [q^p_{i,lo}, q^p_{i,hi}] \cap [q^o_{i,lo}, q^o_{i,hi}] = [q^p_{i,lo}, q^p_{i,hi}]$, and hence $\ell_i(I'_i) = \{[q^p_{i,lo}, q^p_{i,hi}]\} = \iota'_i$.

Now suppose instead that $q_{i,hi} - q_{i,lo} < k\mu\tau$. Then $q_{i,hi} - q_{i,lo} \leq (k-1)\mu\tau$ and we can take $\chi_i = (q^o_{i,lo} + q^o_{i,hi})/2$ and $I_i = [q_{i,lo} + \mu\tau/2 - \epsilon_{lo}, q_{i,hi} - \mu\tau/2 + \epsilon_{hi}] := [x_{i,lo}, x_{i,hi}]$ for ϵ_{lo} and ϵ_{hi} such that $\epsilon_{\min} < \epsilon_{lo} < \mu\tau$ and $\epsilon_{\max} < \epsilon_{hi} < \mu\tau$, yielding $x_{i,hi} - x_{i,lo} = (q_{i,hi} - \mu\tau/2 + \epsilon_{hi}) - (q_{i,lo} + \mu\tau/2 - \epsilon_{lo}) = q_{i,hi} - q_{i,lo} - \mu\tau + \epsilon_{lo} + \epsilon_{hi} < q_{i,hi} - q_{i,lo} + \mu\tau \leq k\mu\tau$. By the same reasoning as in Case 2, $\ell_i(I_i) = \iota_i$, $L_i(\chi) = [q^o_{i,lo}, q^o_{i,hi}]$ and $I^p_i(I, v_c) = [x^p_{i,lo}, x^p_{i,hi}]$, for some $x^p_{i,lo}$ and $x^p_{i,hi}$ such that $q^p_{i,lo} - \mu\tau/2 < x^p_{i,lo} \leq q^p_{i,lo} + \mu\tau/2$ and $q^p_{i,hi} - \mu\tau/2 < x^p_{i,hi} \leq q^p_{i,hi} + \mu\tau/2$. Since $q^o_{i,lo} \leq q^p_{i,lo}$ and $q^o_{i,hi} \geq q^p_{i,hi}$, it follows that $q^p_{i,lo} - \mu\tau/2 < \max\{x^p_{i,lo}, q^o_{i,lo}\} \leq q^p_{i,lo} + \mu\tau/2$ and that $q^p_{i,hi} - \mu\tau/2 < \min\{x^p_{i,hi}, q^o_{i,hi}\} \leq q^p_{i,hi} + \mu\tau/2$. Thus, $I'_i = I^c_i(I^p(I, v_c), \chi) = [x^p_{i,lo}, x^p_{i,hi}] \cap [q^o_{i,lo}, q^o_{i,hi}] = [\max\{x^p_{i,lo}, q^o_{i,lo}\}, \min\{x^p_{i,hi}, q^o_{i,hi}\}]$, so that $\ell_i(I'_i) = \{[q^p_{i,lo}, q^p_{i,hi}]\} = \iota'_i$. □

CHAPTER V

Conclusion & Future Work

5.1 Dynamic Diagnosability

This work considered the problem of dynamic fault diagnosis under the constraint of maintaining K-diagnosability. Using an appropriately defined information state, we constructed a structure called the MPO that contains all the solutions to the problem. Our contributions are as follows. By defining the information state as we have, we reduce the space complexity of the MPO from $O(2^{|X|^2 \cdot K \cdot 2^{|E|}})$ in *Cassez and Tripakis* (2008) to $O(2^{|X| \cdot (K+2)} 2^{|E|})$. Furthermore, the monotonicity property on the extended specification and the resulting reduction of the information state reduces this complexity to $O(2^{|X|}(K+2)^{|X|} 2^{|E|})$, yielding a complexity that is polynomial in K, rather than exponential in K. Finally, the use of the extended specification allows for an efficient quadratic time test for determining the safety of a control decision from any given information state. This potentially allows for a minimal solution to be computed on-line, simply by taking a minimal control decision from each information state (all such controllers will be minimal but not all minimal controllers will have this form).

Implementation of the results of this work on real world systems may require further effort. In particular, the reduction of the size complexity of the MPO and the efficient algorithmic implementation for its construction are unlikely to be able

to deal with systems containing thousands of states, much less millions. Although we have constructed MPOs with millions of states in minutes, those MPOs corresponded to automata that had only a hundred states. The problem here is that the MPO scales exponentially with the size of the automaton to diagnose and therefore its full construction is impractical for real world systems. One way to deal with this issue is to compute minimal solutions on-line, as discussed above. The running time of this is exponential in the number of events, but only quadratic in the number of states of the automaton to diagnose. Since the number of events is typically much smaller than the number of states, this method should scale to much larger automata. A second possible solution to the problem of computationally intractable MPOs is to seek a single controller C that is optimal according to some cost criterion. Indeed, both the works of *Thorsley and Teneketzis* (2007) and *Cassez and Tripakis* (2008) use this approach. In *Cassez and Tripakis* (2008), the construction of the MPO is used as a first step, however. In *Thorsley and Teneketzis* (2007), a dynamic program is solved over a similarly sized structure (essentially an MPO without Z states). In general, the optimal sensor activation policy from any given state depends on that state's successors (and so on...), so that a large portion of the state space must be explored in order to determine the optimal policy. This could potentially be mitigated by heuristics like alpha-beta pruning, since the optimization is of the min-max form.

A possible direction for future work is the extension of this work to the decentralized case of co-diagnosability, in which a number of sites each have a distinct (but possibly overlapping) set of sensors and must jointly diagnose the occurrence of a fault. Specifically, co-diagnosability requires that any fault is eventually diagnosed by at least one site. The problem may be analyzed with or without communication between the various diagnosing sites. It should however be noted that maximally permissive solutions will generally not exist in the decentralized setting, since one site may be required to turn more sensors on if another turns fewer sensors on. Thus,

a feasible solution concept would be either the construction of a minimal sensor activation policy, or the computation of an optimal policy.

5.2 Vehicle Control

We considered the problem of finding a safe, non-deadlocking, and maximally permissive supervisor for a set of vehicles at an intersection, in the presence of uncontrolled vehicles, bounded disturbances, and measurement uncertainty. We showed how to compute the desired supervisor by creating an abstraction, solving for the maximally permissive supervisor in the abstracted domain, and translating back to the original problem domain. Our contributions are threefold. First, in the domain of modelling, we showed how to construct DES abstractions for systems with environmental uncertainty by discretizing the state space, using uncontrollable events to model sources of environmental uncertainty and, in the case of imperfect measurement, using observable but uncontrollable events to model measurement uncertainty. We also showed how to translate safety and marking specifications defined over a continuous state space to the DES domain, yielding a language based specification. Second, we defined new relations between systems and their DES abstractions, allowing for supervisors to be computed using a "abstract-solve-translate" method and characterizing the class of supervisors over which the result will be maximally permissive. Finally, in the case of perfect measurement, we provided customized algorithms for computing the desired supervisors and demonstrated their scalability through simulations. Notably, these algorithms can be applied to the case of imperfect measurement with little modification.

Whereas we have developed solution methods capable of dealing with environmental uncertainty and measurement uncertainty, there remains further work to be done in order to implement these methods on real world systems. One significant issue is that, because we make no assumptions on the behaviour of the uncontrolled vehicles,

their presence severely restricts the set of problem instances that have non-empty supervisors. A possible solution would be to model driver behaviour. Another problem characteristic which restricts the set of problem instances that have non-empty supervisors is the requirement of absolute safety. A possible solution would be to consider a stochastic model (especially for the disturbance) and to require only that the supervisor prevent collisions with some probability $1 - \epsilon$. Finally, a more realistic model of the system dynamics would be as a second order differential equation.

Possible avenues for future work include extensions to a stochastic model or to more complex dynamics, as described above. Another direction for future work is the improvement of scalability. The simulations we have conducted in the case of perfect measurement indicate that memory utilization is the bottle-neck, not running time. Two possibilities to address this issue are the computation of a minimal set of boundary states between safe and unsafe states and the use of dynamic discretization. In this context, dynamic discretization would consist of computing an initial supervisor at a coarse discretization, refining the discretization at states not known to be safe, recomputing a supervisor, and so on... This method could be improved upon by determining the set of safe, unsafe, and uncertain states at each discretization level. Specifically, the supervisor synthesis problem can be thought of as a game against an adversary: the winning states for the controller are safe; the winning states for the adversary are unsafe; and the states that are losing to both controller and adversary are uncertain. Refining the discretization would then be performed only on the uncertain states.

BIBLIOGRAPHY

BIBLIOGRAPHY

Ahn, H., A. Colombo, D. Del Vecchio, and S. Lafortune (2014), Supervisory Control for Intersection Collision Avoidance in the Presence of Uncontrolled Vehicles, in *American Control Conference (ACC), 2014*.

Alur, R., T. Henzinger, G. Lafferriere, and G. Pappas (2000), Discrete abstractions of hybrid systems, *Proceedings of the IEEE*, *88*(7), 971–984, doi:10.1109/5.871304.

Au, T.-C., C.-L. Fok, S. Vishwanath, C. Julien, and P. Stone (2012), Evasion planning for autonomous vehicles at intersections, in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1541–1546, doi:10.1109/IROS.2012.6385936.

Belta, C., A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. Pappas (2007), Symbolic planning and control of robot motion [Grand Challenges of Robotics], *IEEE Robotics Automation Magazine*, *14*(1), 61–70, doi:10.1109/MRA.2007.339624.

Bruni, L., A. Colombo, and D. Del Vecchio (2013), Robust multi-agent collision avoidance through scheduling, in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*.

Camara, J., A. Girard, and G. Gossler (2011), Safety controller synthesis for switched systems using multi-scale symbolic models, in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pp. 520–525, doi:10.1109/CDC.2011.6160424.

Cassandras, C. G., and S. Lafortune (2008), *Introduction to Discrete Event Systems*, 2nd ed., Springer.

Cassez, F., and S. Tripakis (2008), Fault Diagnosis with Static and Dynamic Observers, *Fundamenta Informaticae*, *88*(4), 497–540.

Cassez, F., S. Tripakis, and K. Altisen (2007a), Sensor Minimization Problems with Static or Dynamic Observers for Fault Diagnosis, in *Application of Concurrency to System Design, 2007. ACSD 2007. Seventh International Conference on*, pp. 90–99, doi:10.1109/ACSD.2007.27.

Cassez, F., S. Tripakis, and K. Altisen (2007b), Synthesis Of Optimal-Cost Dynamic Observers for Fault Diagnosis of Discrete-Event Systems, in *Theoretical Aspects of*

Software Engineering, 2007. TASE '07. First Joint IEEE/IFIP Symposium on, pp. 316–325, doi:10.1109/TASE.2007.51.

Cassez, F., J. Dubreil, and H. Marchand (2009), Dynamic Observers for the Synthesis of Opaque Systems, in *Automated Technology for Verification and Analysis*, *Lecture Notes in Computer Science*, vol. 5799, pp. 352–367, Springer Berlin / Heidelberg.

Colombo, A., and D. Del Vecchio (2011a), Enforcing Safety of Cyberphysical Systems Using Flatness and Abstraction, *SIGBED Rev.*, *8*(2), 11–14, doi:10.1145/2000367.2000369.

Colombo, A., and D. Del Vecchio (2011b), Supervisory control of differentially flat systems based on abstraction, in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pp. 6134–6139, doi:10.1109/CDC.2011.6160759.

Colombo, A., and D. Del Vecchio (2012), Efficient Algorithms for Collision Avoidance at Intersections, in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '12, pp. 145–154, ACM, New York, NY, USA, doi:10.1145/2185632.2185656.

Colombo, A., and A. Girard (2013), An approximate abstraction approach to safety control of differentially flat systems, in *Control Conference (ECC), 2013 European*, pp. 4226–4231.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2009), *Introduction to Algorithms*, MIT Press and McGraw-Hill.

Dallal, E., and S. Lafortune (2010), On Most Permissive Observers in Dynamic Sensor Optimization Problems for Discrete Event Systems, in *Proceedings of the 48th Annual Allerton Conference on Communication, Control, and Computing*.

Dallal, E., and S. Lafortune (2011a), A framework for optimization of sensor activation using most permissive observers, in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pp. 2711–2717, doi:10.1109/CDC.2011.6160506.

Dallal, E., and S. Lafortune (2011b), Efficient Computation of Most Permissive Observers in Dynamic Sensor Activation Problems, in *2nd International Workshop on Logical Aspects of Fault-Tolerance (LAFT)*, doi:10.1109/CDC.2011.6160506.

Dallal, E., and S. Lafortune (2014), On most permissive observers in dynamic sensor activation problems, *Automatic Control, IEEE Transactions on*, *59*(4), 966–981, doi:10.1109/TAC.2013.2294613.

Dallal, E., A. Colombo, D. Del Vecchio, and S. Lafortune (2013a), Supervisory control for collision avoidance in vehicular networks using discrete event abstractions, in *American Control Conference (ACC), 2013*, pp. 4380–4386.

Dallal, E., A. Colombo, D. Del Vecchio, and S. Lafortune (2013b), Supervisory control for collision avoidance in vehicular networks with imperfect measurements, in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 6298–6303, doi:10.1109/CDC.2013.6760885.

Dallal, E., A. Colombo, D. Del Vecchio, and S. Lafortune (2014), Supervisory Control of Systems under Imperfect Measurements with Applications to Vehicle Control, *submitted to Discrete Event Dynamic Systems*.

Daws, C., and S. Tripakis (1998), Model checking of real-time reachability properties using abstractions, in *Tools and Algorithms for the Construction and Analysis of Systems*, *Lecture Notes in Computer Science*, vol. 1384, edited by B. Steffen, pp. 313–329, Springer Berlin Heidelberg, doi:10.1007/BFb0054180.

Girard, A., G. Pola, and P. Tabuada (2010), Approximately Bisimilar Symbolic Models for Incrementally Stable Switched Systems, *Automatic Control, IEEE Transactions on*, *55*(1), 116–126, doi:10.1109/TAC.2009.2034922.

Hadj-Alouane, N. B., S. Lafortune, and F. Lin (1994), Variable lookahead supervisory control with state information, *Automatic Control, IEEE Transactions on*, *39*(12), 2398–2410.

Hafner, M., and D. Del Vecchio (2011), Computational Tools for the Safety Control of a Class of Piecewise Continuous Systems with Imperfect Information on a Partial Order, *SIAM Journal on Control and Optimization*, *49*(6), 2463–2493, doi:10.1137/090761203.

Hafner, M., D. Cunningham, L. Caminiti, and D. Del Vecchio (2013), Cooperative collision avoidance at intersections: Algorithms and experiments, *Intelligent Transportation Systems, IEEE Transactions on*, *14*(3), 1162–1175, doi:10.1109/TITS.2013.2252901.

Haji-Valizadeh, A., and K. Loparo (1996), Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems, *IEEE Transactions on Automatic Control*, *41*(11), 1579–1593.

Jiang, S., R. Kumar, and H. Garcia (2003), Optimal sensor selection for discrete-event systems with partial observation, *IEEE Transactions on Automatic Control*, *48*(3), 369–381.

Kowshik, H., D. Caveney, and P. Kumar (2011), Provable Systemwide Safety in Intelligent Intersections, *Vehicular Technology, IEEE Transactions on*, *60*(3), 804–818, doi:10.1109/TVT.2011.2107584.

LaValle, S. M. (2006), *Planning algorithms*, Cambridge university press.

Piterman, N., A. Pnueli, and Y. Sa'ar (2006), Synthesis of reactive (1) designs, in *Verification, Model Checking, and Abstract Interpretation*, *Lecture Notes in Computer*

Science, vol. 3855, edited by E. Emerson and K. Namjoshi, pp. 364–380, Springer Berlin / Heidelberg.

Pola, G., and P. Tabuada (2009), Symbolic Models for Nonlinear Control Systems: Alternating Approximate Bisimulations, *SIAM Journal on Control and Optimization*, *48*(2), 719–733, doi:10.1137/070698580.

Ramadge, P., and W. Wonham (1989), The control of discrete event systems, *Proceedings of the IEEE*, *77*(1), 81–98, doi:10.1109/5.21072.

Ramadge, P. J., and W. M. Wonham (1987), Supervisory control of a class of discrete event processes, *SIAM journal on control and optimization*, *25*(1), 206–230.

Rowaihy, H., S. Eswaran, M. Johnson, D. Verma, A. Bar-Noy, T. Brown, and T. La Porta (2007), A survey of sensor selection schemes in wireless sensor networks, in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 6562, doi:10.1117/12.723514.

Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis (1995), Diagnosability of discrete event systems, *IEEE Transactions on Automatic Control*, *40*(9), 1555–1575.

Tabuada, P. (2009), *Verification and control of hybrid systems: a symbolic approach*, Springer.

Thorsley, D., and D. Teneketzis (2007), Active Acquisition of Information for Diagnosis and Supervisory Control of Discrete Event Systems, *Discrete Event Dynamic Systems: Theory and Applications*, *17*(4), 531–583.

Tomlin, C. J., I. Mitchell, A. M. Bayen, and M. Oishi (2003), Computational techniques for the verification of hybrid systems, *Proceedings of the IEEE*, *91*(7), 986–1001.

Verma, R., and D. Del Vecchio (2011), Semiautonomous multivehicle safety, *Robotics & Automation Magazine, IEEE*, *18*(3), 44–54.

Wang, W., S. Lafortune, and F. Lin (2007), An algorithm for calculating indistinguishable states and clusters in finite-state automata with partially observable transitions, *Systems & Control Letters*, *56*(9-10), 656–661, doi:10.1016/j.sysconle.2007.03.006.

Wang, W., S. Lafortune, F. Lin, and A. Girard (2009), An online algorithm for minimal sensor activation in discrete event systems, in *Proceedings of the 48th IEEE Conference on Decision and Control, CDC/CCC 2009*, pp. 2242–2247, doi:10.1109/CDC.2009.5400888.

Wang, W., S. Lafortune, A. Girard, and F. Lin (2010), Optimal sensor activation for diagnosing discrete event systems, *Automatica*, *46*(7), 1165–1175.

Wonham, W. (2013), Supervisory Control of Discrete-Event Systems, `http://www.control.toronto.edu/people/profs/wonham/wonham.html`.

Wonham, W., and P. Ramadge (1987), On the Supremal Controllable Sublanguage of a Given Language, *SIAM Journal on Control and Optimization*, *25*(3), 637–659, doi:10.1137/0325036.

Yoo, T.-S., and H. E. Garcia (2008), Diagnosis of behaviors of interest in partially-observed discrete-event systems, *Systems & Control Letters*, *57*(12), 1023–1029, doi:10.1016/j.sysconle.2008.06.009.

Yoo, T.-S., and S. Lafortune (2002), NP-completeness of sensor selection problems arising in partially observed discrete-event systems, *IEEE Transactions on Automatic Control*, *47*(9), 1495–1499, doi:10.1109/TAC.2002.802762.

Zamani, M., G. Pola, M. Mazo, and P. Tabuada (2012), Symbolic Models for Nonlinear Control Systems Without Stability Assumptions, *Automatic Control, IEEE Transactions on*, *57*(7), 1804–1809, doi:10.1109/TAC.2011.2176409.

Zaytoon, J., and M. Sayed Mouchaweh (2012), Discussion on fault Diagnosis methods of Discrete Event Systems, in *Proceedings of the 11th International Workshop on Discrete Event Systems (WODES'12)*.