Classical and Logic Based Control Theory for Finite State Machines

Suning Wang

B.Eng. Beijing Institute of Technology, Beijing, China

M.Sc. Academia Sinica, Beijing, China

Department of Electrical Engineering

McGill University, Montréal

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

October 1991

© Suning Wang

Abstract

This thesis formulates the state estimation and control problem for partially observed finite machines in terms of *classical* and *logic-based* approaches. First, in Part I, we present a set operation based formulation of an *observer (tree)* and a dynamic programming based *controller*. Then we provide the results of computational complexity of building and runing such classical observer and controllers. In Part II, we introduce a notion of a logic-based dynamical system, a new paradigm for controlling finite machines. In particular, we give concepts of a *logic-based dynamic observer*, and a *logic-based dynamic controller* and demonstrate an equivalence between classical and logic-based systems. Then we introduce a *conditional observer and controller logic – COCOLOG* for finite machines, which consists of a family of first order logics each corresponding to a node in the observer tree. Conditional control statements are formulated so that (closed loop) control actions occur when specified *past measurable* (i.e. past observation dependent) conditions are fulfilled. A semantics is supplied for each COCOLOG in terms of interpretations of controlled transitions on a tree of state estimate sets indexed by observation $o(k)$. Consistency and completeness of the first order theories in a COCOLOG family are established. Furthermore, through a certain unique model property, we obtain the de dability result for each logical theory in a COCOLOG family. Last, in Part III, a function evaluation based resolution for COCOLOG theorems, called FE–resolution, is presented. Completeness results for the FE–resolution method is given in terms of relative truthfulness and validity.

Résumé

Dans cette thèse sont formulés les problèmes de l'estimation des variables d'état et de la commande de systèmes partiellement observés et finis, en suivant à la fois la démarche classique et une nouvelle démarche basée sur une utilisation de systèmes logiques En première partie, opérant sur des ensembles et reposant sur une programmation dynamique, un observateur (arbre) et un contrôleur sont formulés, suivis d'une évaluation de la complexité de calcul nécessaire pour établir et exploiter ces modèles En deuxième partie, un paradigme permettant de gouverner les systèmes finis, basé sur le concept d'un système dynamique logique, est introduit En particulier, les fondements d'un observateur et d'un contrôleur dynamiques logiques, dont l'équivalence avec les systèmes classiques est démontrée, sont élaborés. Nous introduisons alors une logique définissant un observateur et un contrôleur conditionels, ou COCOLOG, opérant sur des systèmes finis, COCOLOG consiste en une famille de systèmes logiques de premier ordre, chacun correspondant à un noeud dans l'arbre de l'observateur classique. Des énoncés conditionels de gouvernabilité sont formulés, permettant une commande en boucle fermée lorsqu'un passé measurable est observé selon les conditions exigées. Pour chaque système logique de COCOLOG une sémantique interprétant l'action du contrôleur sur les déplacements dans l'arbre à partir des observations o(k) est détaillèe. La cohérence et la récursivité de ces systèmes logiques de premier ordre sont démontrées. De plus, grâce à une propriété d'unicité du modèle, la détermination de chaque système logique de COCOLOG est démontrée. Finalement, une résolution appelée FE-resolution et basée sur une fonction d'évaluation est développée;

une preuve de la complétude de cette résolution est donnée en termes de véracité et de validité.

Claims of Originality

In this thesis the following original contributions are made

- The structural properties of both the current and initial state observer trees for partially observed input-state-output finite machines, together with associated estimates of the complexity of the current and initial state observer trees are presented
- A formulation of the state guidance control problem, for both the completely and partially observed input-state-output finite machines, in terms of the backward recurrence equations of dynamic programming is given
- Associated estimates of the computational complexity of the generation of control laws for completely and partially observed finite machines are presented
- A logic-based paradigm for control theory is formulated this involves, in particular, the conception of a logic-based dynamical system, or a dynamical system of logics.
- COCOLOG (a conditional observer and controller logic) is presented, which is a family of first order logical theories for the state estimation and control of any finite machine.
- The consisteny and completeness results are obtained for each logical theory in a COCOLOG system.

- The concept of FE-resolution - a resolution based theorem proving methodology augmented with the function evaluation (FE-resolution) facility is introduced.
- The completeness result for FE-resolution is established.

Acknowledgement

I would like to thank Professor Peter E. Caines, my thesis supervisor, for his excellent technical supervision during the years of my thesis research and development I would also like to thank him arranging for my financial support throughout my graduate work in McGill, but especially I would like to thank him for his friendship, his constant encouragement and his example of which were absolutely vital for the completion of this thesis

I would like to thank Prof Russell Greiner for his technical support on the issues related to the subject of artificial intelligence I would also like to thank him and Prof Caines as well for the chance to collaborate which produced most of the results reported in the Chapter 2 of this thesis.

I would like to thank Prof M.Makkai for his detailed and constructive comments to the thesis and co-organizing the Logic and Adaptive Control seminars with Prof P E Caines from which I first learnt modal logic and dynamic logic among other things I like to thank Prof. David Delchamps for his suggestions of the acronym of COCOLOG which now becomes the title of our family of logics for logic-based control systems, to thank Geoffrey Hyman for the discussions on the complexity of the resolution proof method of the pigeonhole formula and to thank Profs. Dickinson, Ramadge and Sontag for discussions of and references to the finite automata literature I would like to thank Martin Cote who helped me to translate the abstract into French I am also grateful to the members of the systems and control group at McGill for the stimulating environment

Finally, I would like to express my special thanks to my wife Lan for her love and support and to my parents for their understanding and love over the years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Discrete Event Dynamical Systems

A dynamical system is generally understood to be a system that evolves with time, furthermore, throughout this thesis we shall take the term *system* to refer to a dynamical system which possesses observed or unobserved inputs which may either be manipulated (called controls), not subject to such an influence (generally called disturbances), or be a collection of broad types of input. Conventional *systems and control theory* (SCT) has a long history of handling continuous controlled dynamical systems and their sampled data counterparts, such as aerospace vehicles and manufacture systems. Currently, however, the closed-loop, control theoretic analyses are also emerging for the class of the so called *discrete event dynamical systems*, such systems are becoming increasingly complicated with the massive application of, and fast advances in, computer technology in our century. These human created systems appear to have particular artificial properties that distinguish them from classical dynamical systems – despite the fact that they are evidently manufactured from natural substances. For instance , the flow of statements of an automatic theorem proving algorithm or a machine translation algorithm for natural languages has many significant features and properties beyond the physical flow of the events of the computation: in addition to the intrinsically discontinuous nature of the phenomena (a feature not exhibited by sampled version of continuous systems), such process may be conceived so as to exhibit semantical properties.

The classical dynamical (deterministic or stochastic, linear or nonlinear, finite or infinite, discrete or continuous) systems often arise from systems modeled mathematically by differential or difference equations. Examples of such systems are those governed by classical mechanical laws and the laws of electromagnetism (Newtonian, Lagrangian, Hamiltonian and Maxwell), and indeed chemistry and physics in general The concepts of *controllability, observability, state space realization*, etc were introduced in the late 50's as a part of the fruitful development of *linear system theory* and this led to the successful application of the control theoretic results in the design of many modern systems. Moreover contemporary control theory is partly engaged in the generalization and extension of such ideas to non-linear and stochastic systems.

Discrete event (human created) *dynamical systems* are not governed by the laws of classical mechanics or electromagnetism. These systems are driven by and emit finite or infinite sequences of intrinsically discontinuous events which can occur at possibly unknown irregular spaced instants of time.

Examples of discrete event system structures can be found in computer system software/hardware design and verification, production or assembly lines, queuing systems, communication networks, traffic systems, robotics and expert systems etc , at least, at some level of system modeling. State transition in such a state space model of a system is often called an *event*. Typical events in a real system can be. *time out, message sent, message received* in a communication protocol; *departure* or *arrival* of a transaction in a database; or *switching* between alternative rules in an expert system

There is a vast range of studies of discrete event dynamical systems using modeleds given by finite machines, automata theory, formal languages and classical and modal logic.

Over at least the last twenty years, there has been a widespread effort to give detailed models and formal analyses of the complex systems generated by computer engineering, computer science and its associated technologies. Among the discrete models that deserve mention are communicating sequential processes [Hoa85, MM79, TW86], concurrent program modeling, semantics and correctness verification [Sha78, Pnu79, MP81, Pet81]; synchronization in operating systems [Dij74]; communication protocols, [RCV88, Kur86, Oku88]; digital circuit design and verification, see [GK87, RK87], database concurrency control, [Laf88, LW85].

Recently, the problem has attracted the attention of many systems and control theorists and some significant results under the framework of *supervisory control theory of discrete event dynamical system* have been obtained by M.Wonham and P.Ramadge, [RW87, WR87]. (In the next section we give a brief review of this theory)

Nerode has shown that for any non-anticipative input-output system there exists a dynamical state space realization via the construction of the *Nerode equivalence classes* over the input space, for an explanation of this, see [Cai88] Any discrete event input-output system may generally be taken to be a non-anticipative input-output system and therefore has a state space realization. For such systems we often assume that the *dimension* or the *cardinality* of the state space is finite and that the *state transition* occurs at irregular, distinct discrete time instants. Furthermore the *state value* is often only symbolic, that is to say there is no topology on the state space or the input-output spaces. Figure 1 1 shows the classification over state space systems based on these three criteria, where known mathematical modeling of each class of the state space systems is given in the graph.

In this thesis we shall construct logic control systems for finite machines. We note however that there are clearly many different aspects of discrete event systems which are not modeled by finite machines. Finite machines model the orderly flow of events or states under partial or complete observation of a given system. The timing issue, i.e., when and how the state transition takes place, and the quantitative issue, i.e. those properties

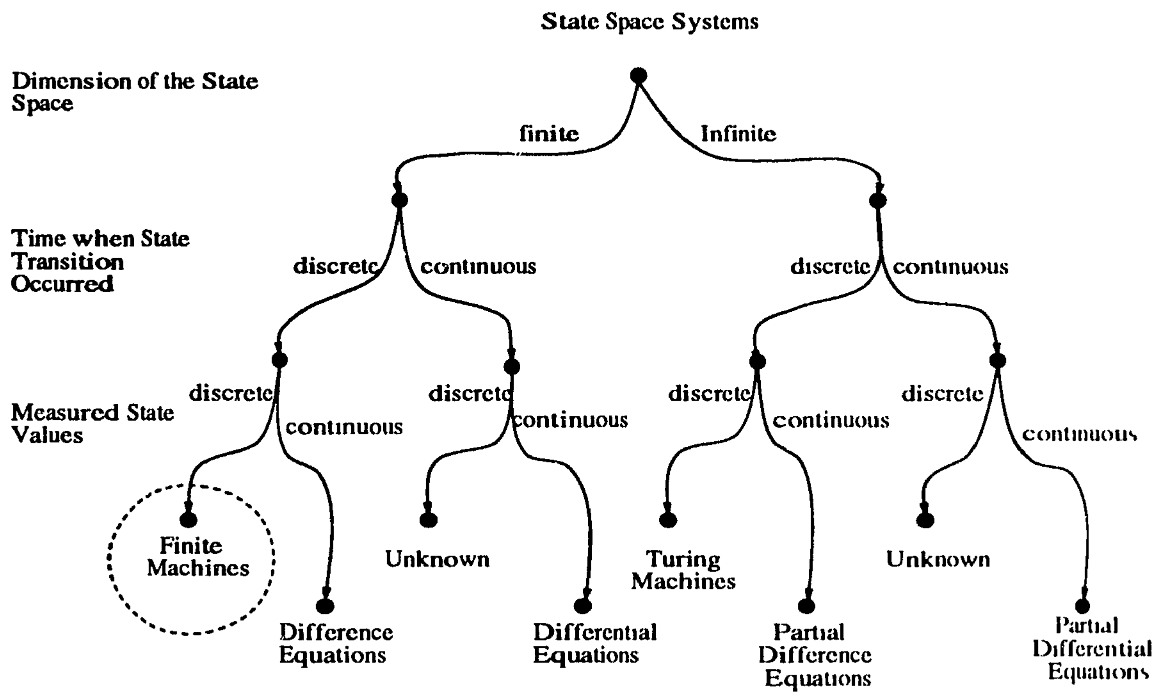


Figure 1.1: Classification of State Space Systems

modeled by queuing theory are examples of properties which are beyond the scope of the standard finite machine framework; for a survey of the modeling of these phenomena see [RW89, Ho87].

1.2 Past Work

In this section we give a brief review of the basic results of the supervisory control theory of Wonham and Ramadge and other related results.

1.2.1 Supervisory Control Theory

Wonham and Ramadge were the first to introduce the closed loop control theoretic framework for the class of discrete event systems, [RW87, WR87]. They showed that the concepts of controllability, observability, state feedback closed loop control, etc. can play important roles in what they called supervisory control problems for the discrete event systems. Recent development of supervisory control theory has been extended to include the issue of aggregation, decentralized, hierarchical, modular and distributed controls.

Supervisory control problem is formulated in terms of automata and formal language. A *plant*, i.e , a physical system to be controlled, is modeled by a formal language $L(G)$, with alphabet or event set Σ, which represents all possible behaviors of the physical system. Some of these behaviors are undesirable. Then a control mechanism is introduced by the partition of the event set Σ into the set of controllable events Σ_c and the set of uncontrollable events Σ_{uc}, such that $\Sigma = \Sigma_c \bigcup \Sigma_{uc}$. A controllable event can be enabled or disabled. In order for a plant to behave as specified, a controller (or a supervisor) $\mathcal{S}$ is constructed to supervise the controllable events The controller is determined as a function of the current state of the plant and the specified behavior of the desired system. The formal languages generated by the closed loop system with regulator $\mathcal{S}$ is denoted $L_c(\mathcal{S}/G)$. To characterize the language that can be generated by a closed-loop structure, the concept of a controllable language is introduced. Given a language L over an alphabet Σ, a prefix closure $\overline{L}$ of L is defined to be $\overline{L} = \{u : uv \in L \text{ for some } v \in \Sigma^*\}$, where Σ^* is the set of all strings from Σ. A language $K \subset L$ is *controllable* if it satisfies $\overline{K}\Sigma_u \bigcap L \subset \overline{K}$. The physical meaning of this definition is that K is controllable if each string from the prefix closure concatenated with an uncontrollable event is in the language L should also be a string of the prefix closure of K.

One of the principal results of [RW87, WR87] is the following: a supervisor $\mathcal{S}$ exists such that $L_c(\mathcal{S}/G) = K$ for some desirable behavior specified by K if and only if K is controllable and $L_m(G)$ is closed, i.e., K is driven from any marked state of $L(G)$ to some marked state of $L(G)$. Where $L_m(G)$ is a subset of $L(G)$ denotes the set of marked strings. Furthermore, the supervisor $\mathcal{S}$ can be generated so as to give the maximum permissible behavior or, equivalently, the minimally disabled system that satisfies the given specifications.

1.2.2 Extensions to Supervisory Control Theory

The supervisory control framework has been extended since Wonham and Ramadge initiated their study. In this subsection we mention some of these extensions

The supervisory control framework depends on the complete observation of the events in a plant for the supervisor to provide controls In a decentralized and hierarchical situation, or with the presence of noise in the observations, a partial observation based control solution is necessary. Cieslak, et al [RCV88] introduce an observation function which maps the events into a set of output symbols. The supervisory control task is then based on observations of the output string Lin and Wonham [LW87] also consider the problem of constructing the supervisor based on the presence of an event observer for a partially observed plant.

Ostroff and Wonham [Ost89, OW89a, OW85] have extended the automata and formal language model to include a time window within which each transition takes place. Having this time window, the automata model is then able to describe the timing of plant behavior These augmented automata models were calledc extended state machines Ostroff and Wonham have also adopted a temporal logic framework, with its semantics designed to match a given extended state machine, so as to verify the behavior of the given machine It is worth pointing out that the role of temporal logic in their model is restricted to that of a formal tool for verification of the correctness of a given extended state machine and its supervisor as a closed loop system. The correctness property is used in the sense of characterizing the desirable closed loop behavior. This behavior includes time constraints and such standard problems as *liveness, safety, deadlock free, fairness* etc addressed in computer program correctness verification [MP81].

Other developments in supervisory control theory include stability and stabilizability as introduced by Ozveren and Willsky [Ozv89, OW89b]. They give conditions for finding the supremal set such that the marked states are visited from all other states in a finite number of steps. Furthermore, Zhong and Wonham [ZW88] characterize the hierarchical control of discrete event systems.

To summarize, supervisory control theory has provided, for the first time, a systematic synthesis of a closed loop solution to certain qualitative problems in discrete event system theory. Although these problems had previously been attacked in the fields of computer science and data communication, closed loop solutions were not generally obtained.

In the next section, the organization of this thesis along with its main contributions to and relations with current developments of control theoretic solutions to discrete event system problems is presented.

1.3 Organization and Main Contribution

As we illustrated in Figure 1.1, discrete event systems are generally modeled by finite state machines or equivalent mathematical tools. In the rest of this thesis, we shall take finite machines to be the mathematical models of the system in which we are interested. The control of a finite machine is realized through the input function and that of an automaton via its forced transitions. This is equivalent to the control mechanism which enables and disables events due to the application of the forced input.

1.3.1 Observers and Controllers

In Part I of this thesis, we present the basic framework in which our state estimation and control problem is introduced. State estimation is the task of a (current or initial) state observer. In the formulation which we adopt, a controller takes as input the output of a state observer and hence generate the next control to the plant in order to achieve a certain desired behavior of the overall closed-loop system.

We formulate the initial and current state observers for a given finite machine in terms of set-based operations realized by classical observers These observers can also be expressed as tree structures, which consequently are called initial and current state observer trees. We give a set of results on the structural properties of these trees and the size of these trees.

The construction of the controllers described in Part I is formulated in terms of dynamic programming applied to the set of state estimate sets (equivalently the nodes) in the observer trees of a given finite machine. First, the control problem is defined to be that of steering a state (estimate) to a target state. The controller is designed for both complete observation and partial observation cases. A concept of a good state (or state estimate set) is introduced Then system controllability and the existence of a closed-loop controller are determined via backward recurrence equations which, in the partially observed case, will involve the state estimate sets Computational complexity of running and constructing such controllers is also presented. This work should be viewed as foundational for the rest of the thesis and as providing a mathematically well defined framework for the work that follows.

1.3.2 Logic-based Dynamical System and COCOLOG

The formulation of the notion of a logic-based dynamical system and the conditional observer and controller logic-COCOLOG, presented in Part II, is considered to be the main contribution of this thesis.

The procedural and declarative approaches are two basic formulations used for problem solving. These two approaches also draw a line between the methodologies used in computer science and those in artificial intelligence The fundamental feature of the *declarative approach*, or the *logic-based approach*, is that of the immense flexibility of the underlying structure compared with that of the procedural approach This flexibility comes from the expressibility, or richness, of logico–linguistic methods in general, and of first order languages in particular; furthermore, within logic programming it follows from the relative separation of logic and control, where a uniform control mechanism (called an *inference engine* which often consists of *unification* and *resolution algorithms*) is in mechanical theorem proving techniques. In this thesis, we show that the control problems that arise from finite machines can be expressed in terms of those *higher level* descriptions in the sense that they can be generally described by families of first order logics indexed by time The evolution of such logics in time is described by our notion of logic-based dynamical systems, see Chapter 4

Our *conditional observer and controller logic*, called *COCOLOG*, is then defined to express systems and control problems for finite machines and in particular to formulate the problem of designing observers and controllers for a partially observed machines COCOLOG is formulated in terms of a tree of first order logics where each logic corresponds to a node in the observer tree. Among other results we show the consistency and completeness of each COCOLOG logic, a unique model property and thus the decidability property

1.3.3 Automated Reasoning with FE-Resolution

In Part III, we discuss the automated theorem proving for COCOLOG theorems. A function evaluation based resolution, called *FE-resolution*, is presented for the conditional observer and controller logics FE-resolution can also be applied to other general systems. Finally, the completeness of FE-resolution is established. This connects the semantical notion of relative truthfulness to the syntactic procedures of FE-resolution.

1.3.4 On the Tractability of Logic-Based Control Theory for Finite Machines

It is evident that the question of the computational tractability of a formal system such as COCOLOG vitaly effects the realizability, the efficiency of any implementation and, in fact, the possibility of any computational implementation whatsoever.

As we mentioned at the beginning of section 1.3.2, the procedural and declarative formulations are the two basic mechanisms for the implementation of formal computational schemes. At one extreme AI systems are formulated using the declarative formal logico–linguistic systems as a modeling formalism. The critical problem of having such flexibility is the issue of computational tractability associated with a general logico-linguistic systems It is almost invariably the case that the resulting computations are intractable whenever a non-trivial problem is formulated in this framework

On the other hand, if a system is modeled within the framework of conventional mathematics, that is to say, by a procedure or a set of functions, it is often the case that the computational load is tractable but flexibility is lost This is particularly significant in the formulation of high level control, since no general terms, logical quantifiers or alterable axioms can enter the systems in this case Our intention here is to formulate a well defined logical construction for the system theory of finite machines which, in particular, permit the analysis of the computational tractability of the solution of control problems posed at different levels of generality for more or less complex machines. In this way we intend to investigate a flexible, but highly structured, middle ground between the unstructured intractable systems on one hand, and totally structure, rigid and tractable systems on the other.

Part I

Classical Observation and Control Theory for Finite Machines

In this part of the thesis, we shall first, in Chapter 1, take the simple class of dynamical systems represented by partially observed finite machines. We then pose the state estimation problem in terms of the problem of constructing a *classical dynamical system* which generates a sequence of state estimates. Following this, we present the design and complexity analysis of building such an observer for a given finite machine. In Chapter 3, we discuss the problem of constructing a classical dynamical system, which we call a classical dynamical controller, that generates a sequence of controlled inputs. The controllers discussed here are obtained from control laws that solve certain reachability dynamic programming problems: subject to certain conditions these will steer the state estimates to the desired targeted state when the underline finite machine is not completely observed. The question of the complexity of building and running such a controller is covered in the same chapter.

Chapter 2

Dynamical Observers for Finite Machines

In this chapter, we take the class of dynamical systems represented by partially observed finite machines and then pose the state estimation problem in terms of the problem of constructing a classical dynamical system which generates a sequence of state estimates. We first introduce some system concepts which formulate the state observation problem. Then we present a set valued recursive formulation of the observation sets. A representation of these observation sets in tree structures forms the so called *observer trees* for current and initial state observation tasks. Finally a complexity analysis and some structural aspects of the construction of these trees are presented. The contents of this chapter resulted from joint works with Dr. P.E. Caines and Dr. R. Greiner, reported in [CGW91].

2.1 State Observation Problem

A *machine* is taken to mean a deterministic input-state-output machine, a machine is termed *finite* when the input, state and output sets are of finite cardinality. (We will, in general, omit the *input-state-output* qualification.) The dynamical observers for finite machines discussed in this section are themselves modeled by finite machines.

Definition 2.1.1 (Finite Machine) A *finite machine* is a quintuple $\mathcal{M} = (X, U, Y, \Phi, \eta)$ where

X is a (finite) set of *states,*

U is a (finite) set of *inputs,*

Y is a (finite) set of *outputs,*

$\Phi : X \times U \rightarrow X$ is a *transition function,*

$\eta : X \rightarrow Y$ is an *output function.* □

Concerning the notation used in this thesis, we shall sometimes write (i) $\Phi(x, u)$ as $\Phi_u(x)$, and (ii) u_i^n for the $(n - i + 1)$-element sequence $[u_i, u_{i+1}, u_{i+2}, \cdots, u_n]$, where $u_j \in U$ denotes the input at the time instance $j \in \mathcal{Z}_+$ (and where u_j is identified with $[u_j]$) and ϕ^u denotes the empty input; the same notation will be used for x_i^n and y_i^n.

The dynamical evolution of a finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ can be displayed by taking U^* to be the set of all finite sequence of inputs and by extending $\Phi : X \times U \rightarrow X$ to $\Phi : X \times U^* \rightarrow X$, where for all $i, n \in \mathcal{Z}_{+}$, for all $u_i^n \in U^*$ and for all $x \in X$, Φ is recursively defined as:

$$\begin{aligned} \Phi(x, \phi^u) &= x \\ \Phi(x, u_i^n) &= \Phi(\Phi(x, u_i), u_{i+1}^n) \end{aligned} \tag{2.1}$$

Because η is not necessarily a one-to-one map, a finite machine will often be referred to as a *partially observed finite machine.* (The finite machine set-up described above includes, as a special case, that of any *conventional* deterministic finite machine with *partial* state transition function and state output function.)

A finite machine can also be defined by a state transition diagram as shown in Figure 2.1 for a particular seven state machine $\mathcal{M}_7$.

2.1.1 Initial and Current State Observation

The *initial* (respectively, current) *state dynamical observer problem* for a finite machine, $\mathcal{M}$, is to estimate $\mathcal{M}$'s initial (respectively, current) state from observations on its inputs and outputs over a finite time period. An *initial state dynamical observer* takes, as input, the observed behavior of a system, *i.e.*, a sequence of input/output pairs, and outputs an estimate of the initial state of the system.

We can state this formally by the following definitions:

Definition 2.1.2 (N-consistency) The N-element state sequence $x_1^N \in X^N$ is a *N-consistent state sequence* with respect to a given Input/Output sequence, $[\langle u_0, y_1\rangle, \langle u_1, y_2\rangle, \ldots, \langle u_{N-1}, y_N\rangle] \in Y \times (U \times Y)^{N-1}$ if the relation $CS([\langle u_{i-1}, y_i\rangle]_{i=1}^N, x_1^N)$ defined as,

$$x_k = \Phi(x_1, u_1^{k-1}) \text{ and } y_k = \eta(x_k), \quad \text{for all } k \in [1, \cdots, N]. \tag{2.2}$$

is satisfied. The set of all such sequences is denoted $CSS(o_1^N)$. □

(The diagram below illustrates how the x_is, u_is and y_is are related.

$$\begin{tikzcd} \Phi: & {} \arrow[r, "u_0"] & x_1 \arrow[r, "u_1"] \arrow[d] & x_2 \arrow[r, "u_2"] \arrow[d] & x_3 \arrow[d] & \cdots \\ \eta: & & y_1 & y_2 & y_3 & \cdots \end{tikzcd}$$

Where u_0 is usually taken to be the null element denoted by ϕ.)

In other words, an N-consistent state sequence with respect to the input/output sequence $[\langle u_{i-1}, y_i\rangle]_{i=1}^N$, is a *trajectory* of states that satisfies the system dynamics and the observed output sequence from 1 up until time N. We denote by P_k as the projection operator over a sequence of elements defined by $P_k(Z_1, Z_2, \cdots, Z_k, Z_{k+1}, \cdots) = Z_k$ and $O^l \subseteq U^{l-1} \times Y^l$, where $o_1^l \in O^l$ is an observation sequence $[\langle u_{i-1}, y_i\rangle]_{i=1}^l$ of length l generated by some $x \in X$ taken as the initial state together with the corresponding input sequence $u_0, \cdots, u_{l-1}$.

Notice that $CS(\cdot,\cdot)$ is not a function, *i.e.*, there can be many *state sequences* which are consistent with a given I/O sequence, $o_1^k = [\langle u_{i-1}, y_i\rangle]_{i=1}^k$, and that each such state sequence, ${}^j x_{i=1}^k$, is uniquely determined by its first element, ${}^j x_1$. We define the *initial state estimate* as the set of these possible initial states.

Definition 2.1.3 (Initial State Estimate) An *initial state estimate set*, with respect to the N-element observation sequence, o_1^N, written $ISE(o_1^N)$ or $\{\widehat{x_1}\}(o_1^N)$, is the set of initial elements of the consistent state sequences:

$$ISE(o_1^N) \equiv \{\widehat{x_1}\}(o_1^N) = \left\{ x \in X \;\middle|\; \begin{array}{l} x = P_1(x_1^N) \text{ for any } x_1^N \\ \text{such that } x_1^N \in CSS(o_1^N) \end{array} \right\} \tag{2.3}$$

□

In other situations, we may want an estimate of the *current state* of the device, given a sequence of N input/output pairs: $o_1^N = [\langle u_{i-1}, y_i\rangle]_{i=1}^N$

Definition 2.1.4 (Current State Estimate) A *current state estimate set*, with respect to the N-element observation sequence, o_1^N, written $CSE(o_1^N)$ or $\{\widehat{x_N}\}(o_1^N)$, is the set of final elements of the consistent states:

$$CSE(o_1^N) \equiv \{\widehat{x_N}\}(o_1^N) = \left\{ x \in X \;\middle|\; \begin{array}{l} x = P_N(x_1^N) \text{ for any } x_1^N \\ \text{such that } x_1^N \in CSS(o_1^N) \end{array} \right\} \tag{2.4}$$

□

A state-output finite machine will be taken to be an I-S-O finite machine where the input set only contains a single element, *i.e.*, each input is taken to be the same, *e.g.*, a clock tick, u.

Example 2.1.1 We can illustrate these definitions with the state-output 7 state machine, $\mathcal{M}_7$, shown in Figure 2.1. Suppose the device begins in the state x2 Of course, our dynamical observer cannot be a function of any information stating this. Instead, it has access only to the output sequence

$$[\mathrm{y1},\ \mathrm{y2},\ \mathrm{y1},\ \mathrm{y2},\ \ldots]$$

Initially, at time T=0, the initial state observer must consider all states, meaning

$$ISE([\,]) \equiv \widehat{\{x_1\}}(o_1^0) = \{\,\mathrm{x1},\ \mathrm{x2},\ \mathrm{x3},\ \mathrm{x4},\ \mathrm{x5},\ \mathrm{x6},\ \mathrm{x7}\,\}$$

as a possible initial state estimate.

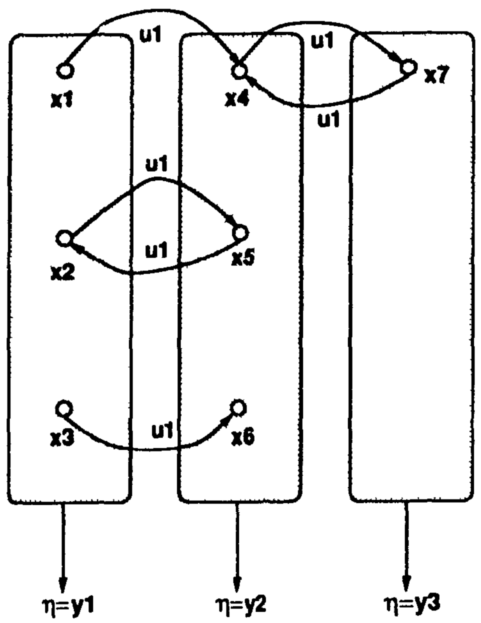


Figure 2.1: The $\mathcal{M}_7$, a state-output finite machine

At time T=1, the initial state observer would output

$$ISE([\mathrm{y1}]) \equiv \widehat{\{x_1\}}(o_1^1) = \{\,\mathrm{x1},\ \mathrm{x2},\ \mathrm{x3}\,\};$$

at time T=2, it outputs the same 3-element set

$$ISE([\mathrm{y1}, \mathrm{y2}]) \equiv \widehat{\{x_1\}}(o_1^2) = \{\,\mathrm{x1},\ \mathrm{x2},\ \mathrm{x3}\,\}.$$

However, at time T=3, it converges to the singleton

$$ISE([\mathrm{y1},\ \mathrm{y2},\ \mathrm{y1}]) \equiv \widehat{\{x_1\}}(o_1^3) = \{\,\mathrm{x2}\,\},$$

which is the same value that it outputs at time T=4

$$ISE([\mathrm{y1},\ \mathrm{y2},\ \mathrm{y1},\ \mathrm{y2}]) \equiv \widehat{\{x_1\}}(o_1^4) = \{\,\mathrm{x2}\,\},$$

and so on: $ISE(n)[\mathrm{y1},\ \mathrm{y2},\ \mathrm{y1},\ \mathrm{y2}, \ldots, \mathrm{yn}] = \{\,\mathrm{x2}\,\}$.

A current state observer would give

$$CSE([\,]) \equiv \widehat{\{x_0\}}(o_1^0) = \{\,\mathrm{x1},\ \mathrm{x2}, \cdots, \mathrm{x7}\,\}$$

$$CSE([\mathrm{y1}]) \equiv \widehat{\{x_1\}}(o_1^1) = \{\,\mathrm{x1},\ \mathrm{x2},\ \mathrm{x3}\,\}$$

$$CSE([\mathrm{y1},\ \mathrm{y2}]) \equiv \widehat{\{x_2\}}(o_1^2) = \{\,\mathrm{x4},\ \mathrm{x5},\ \mathrm{x6}\,\}$$

$$CSE([\mathrm{y1},\ \mathrm{y2},\ \mathrm{y1}]) \equiv \widehat{\{x_3\}}(o_1^3) = \{\,\mathrm{x2}\,\}$$

$$CSE([\mathrm{y1},\ \mathrm{y2},\ \mathrm{y1},\ \mathrm{y2}]) \equiv \widehat{\{x_4\}}(o_1^4) = \{\,\mathrm{x5}\,\}.$$

□

A finite machine is *initial state observable* if we can, eventually, determine its initial state based on observations of its input/output sequence. Likewise, a finite machine is *current state observable* if we can, eventually, determine its current state based on observations of its input/output sequence

In fact we have the following set of definitions of observability for finite machines.

Definition 2.1.5 (Weakly Initial State Observable) A finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is said to be *weakly initial state observable* if for any $x \in X$ there exists a minimum length observation sequence $o_1^{k(x)} \in O^*$, such that for all $N > k(x)$, the initial state estimate set $\widehat{\{x_1\}}(o_1^N) = \{x\}$ is a singleton. □

That is, whatever state the system starts from there always exists an (input-output) observation sequence such that the initial state estimate set contains a single value, after a finite time period.

Definition 2.1.6 (Non-Anticipatively Initial State Observable) A finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is said to be *non-anticipatively initial state observable* if there exists a sequence of non-anticipative input functions $\{u_l : O^{l-1} \to U,\ l \geq 1\}$ i.e., a (non-anticipative) control law u^{NA}, and a constant $K \in \mathcal{Z}_+$, such that for all $x \in X$, and for all $N \geq K$ the initial state estimate $\widehat{\{x_1\}}(o_1^N)$ is a singleton, whenever o_1^N is the output resulting from the input u^{NA}. □

In other words, there exists a past-dependent control law which will force the initial state estimate to give a single value after a finite time period.

Definition 2.1.7 (Strongly Initial State Observable) A finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is said to be *strongly initial state observable* if there exists a $K \in \mathcal{Z}_+$, such that for all $N \geq K$, and all $o_1^N \in O^N$, the initial state estimate $\widehat{\{x_1\}}(o_1^N)$ is a singleton. □

Which is to say that the initial state estimate to be single valued, for any observations after finite time period, which is necessarily the *correct* initial state.

Definition 2.1.8 (Weakly Current State Observable) A finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is said to be *weakly current state observable* if for any $x \in X$ there exists $o_1^{k(x)} \in O^*$ such that for all $N \geq k(x)$, the current state estimate $\widehat{\{x_N\}}(o_1^N) = \Phi(x, u_1^{N-1})$ is a singleton . □

Note here x_N is the *correct* current state, *i.e.*, the correct initial state propagated through the observed inputs.

Definition 2.1.9 (Non-Anticipatively Current State Observable) A finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is said to be *non-anticipatively current state observable* if for all $x \in X$, there exists a sequence of non-anticipative input functions $\{u_l : O^{l-1} \to U,\ l \geq 1\}$, *i.e.*, a (non-anticipative) control law u^{NA} and a constant $K \in \mathcal{Z}_+$, such that for all $N \geq K$, the current state estimate $\widehat{\{x_N\}}(o_1^N) = \Phi(x, u_1^{N-1}) = \{x_N\}$ is a singleton, whenever o_1^N is the output resulting from the input u^{NA}. □

That is, there exists a past-dependent control law which will force the current state estimate to be single valued after a finite time period.

Definition 2.1.10 (Strongly Current State Observable) A finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is said to be *strongly current state observable* if there exists a $K \in \mathcal{Z}_+$, such that for all $N \geq K$, and all $o_1^N \in O^N$, the current state estimate $\{\widehat{x_N}\}(o_1^N)$, is a singleton . □

Theorem 2.1.1 (Weakly iff Non-Anticipatively Observable) For any input-state-output finite machine, $\mathcal{M} = (X, U, Y, \Phi, \eta)$ we have:

(i) If $\mathcal{M}$ is *strongly* initial (respectively cu ent) state observable Then $\mathcal{M}$ is *weakly* initial (respectively current) state observable.

(ii) $\mathcal{M}$ is *weakly* initial (respectively current) state observable if and only if $\mathcal{M}$ is *non-anticipatively* initial (respectively current) state observable.

Proof

(i) is self-evident; further, in (ii) the case of initial state estimation follows from a similar but simpler argument than that of current state estimation. Consequently we shall only prove (ii) here for the current state observable case.

Consider the weakly current state observable finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$, we need to find a family of input functions $\{u_l : O^{l-1} \to U, \, l \geq 1\}$, which will force the current state estimate is single valued after a finite time period.

Such a past-dependent control law can be constructed in the following manner. Since $\mathcal{M}$ is weakly current state observable, there exists $o_1^{k(x)} \in O^*$ for any $x \in X$ such that $u(o_1^N)$ will force $\{\widehat{x_N}\}(o_1^N) = \Phi(x, u_1^{N-1}) = \{x_N\}$ for any $N \geq k(x)$. Therefore we define the first segment of our input function u^{NA} by taking the control actions $u_1^{k(x_1)} \in U^*$ for some arbitrary $x_1 \in X$. This control sequence is used until (1) the current state estimate converges to the singleton in $k(x_1)$ steps which is less than or equal to $|X|$ (see Theorem

3.2.1 in Chapter 3), or (2) an output observation y_p, $1 \leq p \leq k(x_1)$ is obtained which is inconsistent with $o_1^{k(x_1)}$.

In the first case the non-anticipative control sequence u^{NA} has given a sequence of estimates that has converged to a singleton, in the latter, a new control sequence in $o_1^{k(x_2)}$ is initiated at p_1 for some arbitrary $x_2 \in X - \Phi(x_1, u_1^{p-1})$; and (part of) this will form the second segment of u^{NA}. Since $X - \Phi(x_1, u_1^{p-1})$ has $|X| - 1$ elements there are at most $|X| - 1$ elements in the image set $\Phi(X - \Phi^{p_1}(x_1, u_1^{p_1-1}), u_{p_1}^{p_1+k(x_2)-1})$ under the newly chosen control sequence $u_{p_1}^{p_1+k(x_2)-1}$.

Continuing in this manner for $X - \{x_1, x_2\}$, etc, yields (one set of values of) a non-anticipative input function u^{NA} which will take at most $\frac{|X|(|X|+1)}{2}$ steps before the current state estimate converges to a singleton. Performing this process for all possible initial conditions for $\mathcal{M}$ yields a set of sets of values that constitute a non-anticipative control law $u^{NA}(o^*) \to u^*$ satisfying Definition 2.1.6. □

In the rest o, *this thesis we shall restrict discussion solely to the situation where observability is taken in the strong sense*

The relationship between initial and current state observability among finite machines is described by the following example.

Example 2.1.2 In Figure 2.2, we show a two state, state-output finite machine is current state observable but not initial state observable. In general, initial state observability implies the current state observability in each of the classes of weak, non-anticipative and strong, but not vice versa.

The general relationship of current and initial state observability among finite machines is shown in Figure 2.3 in terms of set containment among the set of all possible finite machines.

□

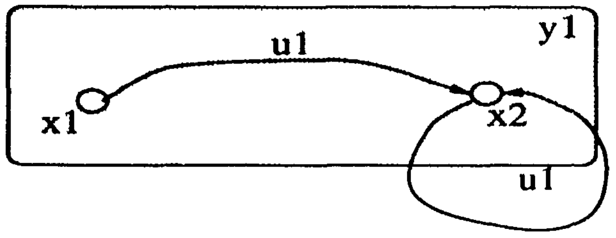
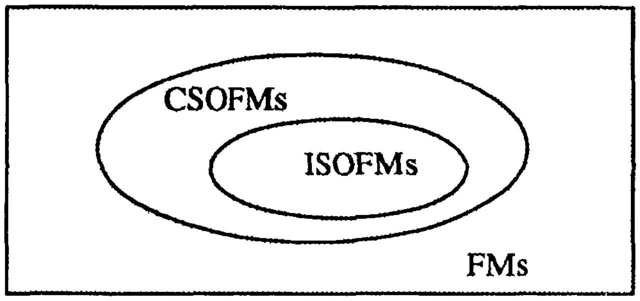


Figure 2.2: A two state machine $\mathcal{M}_2$ is current but not initial state observable



FMs=Finite Machines

CSOFMs=Current State Observable Finite Machines

ISOFMs=Initial State Observable Finite Machines

Figure 2.3: A Nesting With Respect to Observability Properties

Theorem 2.1.2 A finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ *is initial state observable* if and only if the following holds:

$$\exists K \in \mathcal{Z}_+,\ \forall N > K,\ \forall x_0', x_0'' \in X,\ \forall u_1^N \in U^N.$$

$$\bigwedge_{j=1}^{N} [\eta(\Phi(x_0 u_1^j)) = \eta(\Phi(x_0'', u_1^j))] \implies x_0' = x_0''$$

and $\mathcal{M}$ is *current state observable* if and only if

$$\exists K \in \mathcal{Z}_+,\ \forall N > K,\ \forall x_0', x_0'' \in X,\ \forall u_1^N \in U^N.$$

$$\bigwedge_{j=1}^{N} [\eta(\Phi(x_0', u_1^j)) = \eta(\Phi(x_0'', u_1^j))] \implies \Phi(x_0', u_1^N) = \Phi(x_0'', u_1^N)$$

Proof of this theorem is obtained directly from the Definitions 2.1.7 and 2.1.10.

Two states $x_0', x_0' \in X$ of a finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ are said to be *(observability) equivalent*, denoted by $x_0' \equiv_{ob} x_0''$, if and only if for all $u \in U^*, \eta(\Phi(x_0', u)) = \eta(\Phi(x_0'', u))$

The equivalence relation $\equiv_{ob}$ over the state space X of a finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ induces a partition over the equivalence classes of $X/\equiv_{ob}$ which permits us to define another property for finite machines. A finite machine is said to be in *observability reduced form* if and only if

$$\forall x \in X,\ [x]_{\equiv_{ob}} = \{x\}$$

Clearly a finite machine is in observability reduced form if and only if it is initial state observable.

2.1.2 Recursive Set Formulas for State Estimation

We can now give some interesting and useful formulas for the initial and current state estimate sets It will be noted that the Equation 2.6 has the important *predictor-corrector* form of many recursive algorithms in systems and control theory. In fact, in this case, we may refer to the recursion as a *predictor-refiner* formula, since no error-*correction* in the usual sense of the words takes place.

Theorem 2.1.3 ([CGW91] Observation Sets) *Consider any finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$, then for any observation sequence, $o_1^N \in O^N$, the following equations hold:*

$$ISE(o_1^{N+1}) \equiv \widehat{\{x_1\}}(o_1^{N+1}) = \widehat{\{x_1\}}(o_1^N) \bigcap \Phi^{-1}(\eta^{-1}(y_{N+1}), u_1^N) \tag{2.5}$$

$$CSE(o_1^{N+1}) \equiv \widehat{\{x_{N+1}\}}(o_1^{N+1}) = \Phi(\widehat{\{x_N\}}(o_1^N), u_N) \bigcap \eta^{-1}(y_{N+1}) \tag{2.6}$$

These equations may be written as:

$$ISE(o_1^N) \equiv \widehat{\{x_1\}}(o_1^N) = \bigcap_{k=1}^{N} \Phi^{-1}(\eta^{-1}(y_k), u_1^{k-1}) \tag{2.7}$$

$$CSE(o_1^N) \equiv \widehat{\{x_N\}}(o_1^N) = \bigcap_{k=1}^{N} \Phi(\eta^{-1}(y_k), u_k^{N-1}) \tag{2.8}$$

where

- Φ has been extended to take a *set* of states as its first argument:
  $\Phi : \mathcal{P}(X) \times U^* \mapsto \mathcal{P}(X)$ where $\Phi(A, u_1^k) = \{x \in X \,|\, x = \Phi(x', u_0^k)$ for some x' in $A\}$,
- Φ^{-1} is the inverse of Φ: $\Phi^{-1} : \mathcal{P}(X) \times U^* \mapsto \mathcal{P}(X)$ given by
  $\Phi^{-1}(A, u_1^k) = \{\, x \in X \,|\, \Phi(x, u_1^k) \in A \,\}$,
- η^{-1} is the inverse of η: $\eta^{-1} : Y \mapsto \mathcal{P}(X)$ given by
  $\eta^{-1}(y) = \{\, x \in X \,|\, \eta(x) = y \,\}$, and
- $\widehat{\{x_1\}}(o_1^0)$ is defined to be X.

(Here, $\mathcal{P}$ is the power set operator: $\mathcal{P}(B)$ refers to the power set of the set B)

Proof

We first show how the N-consistent states are related to $N+1$-consistent states, then use this to derive the eq[illegible]s shown above. Consider any N-element state sequence, ${}^jx_1^N = [{}^jx_1, {}^jx_2, \cdots, {}^jx_N,]$, which is consistent with an N-long observation sequence, o_1^N, *i.e.*, which satisfies $CS(o_1^N, {}^jx_1^N)$, as specified in Equation 2 2. Now consider the effects on a new observation, $\langle u_N, y_{N+1}\rangle$. If $\Phi({}^jx_N, u_N) = {}^jx_{N+1}$, add it to the end of the ${}^jx_1^N$ sequence, forming ${}^jx_1^{N+1}$. If $\eta({}^jx_{N+1}) = y_{N+1}$, then Equation 2 2 guarantees that $CS(o_1^{N+1}, {}^jx_1^{N+1})$ must hold Notice, further, that this construction accounts for *all* $N+1$-consistent state sequences. Since, for any initial N-element subsequence ${}^kx_1^N$ of any

$N+1$ state sequence ${}^k x_1^{N+1}$ which satisfies $CS(o_1^{N+1}, {}^k x_1^{N+1})$, the relation $CS(o_1^N, {}^k x_1^N)$ must hold.

We prove Equation 2.5 by induction. When $N = 0$, *i.e.*, before any observations, any state is consistent with the empty observation, as $CS([], [x])$ is true for each $x \in X$. Hence Equation 2.3 means $ISE([]) = X$, which is the value specified above.

Now assume Equation 2.5 holds for all $k \leq N$, and we observe $o_{N+1} = \langle u_N, y_{N+1}\rangle$. We saw above that we can expand each N-consistent state sequence ${}^j x_1^N$ into an $N+1$-consistent state sequence ${}^j x_1^{N+1}$, which will qualify as an $N+1$-consistent state only if $\eta(\,\Phi({}^j x_N, u_N)\,) = y_{N+1}$. Hence, an initial element, ${}^j x_1$, which qualified after N observations, *i.e.*, for which ${}^j x_1 \in ISE(o_1^N)$ will remain in $ISE(o_1^{N+1})$ only if $\eta(\,\Phi({}^j x_N, u_N)\,) = y_{N+1}$. Re-expressing this condition in terms of ${}^j x_1$ and u_1^N, gives ${}^j x_1 \in \Phi^{-1}(\,\eta^{-1}(y_{N+1}), u_1^N\,)$. This leads immediately to Equation 2.5. Repeated substitution for the term $ISE(o_1^N)$ yields Equation 2.7.

The proof of Equation 2.6 is similar. The $N = 0$ base step is identical. For the inductive part: Given that ${}^j x_n$ is a member of $CSE(o_1^N)$, we need only observe that ${}^j x_{N+1} = \Phi({}^j x_N, u_N)$ belongs in $CSE(o_1^{N+1})$ if and only if $\eta({}^j x_{N+1}) = y_{N+1}$. Hence, an $x \in X$ should be a member of $CSE(o_1^{N+1})$ if and only if $x = \Phi(x', u_N)$ for some $x' \in CSE(o_1^N)$ *i.e.*, $x \in \Phi(CSE(o_1^N), u_N)$, and $x \in \eta^{-1}(y_{N+1})$. This is exactly Equation 2.6. Equation 2.8 follows immediately. □

The rest of the chapter deals with the design and in addition, a complexity analysis of the design, of a *dynamical observer* for finite machines. First, we provide the relevant type information:

Definition 2.1.11 (Classical Observer System – Type Information) Given any finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$, a *classical observer system* of $\mathcal{M}$ is a finite machine $\widehat{\mathcal{M}} = (\widehat{X}, \widehat{U}, \widehat{Y}, \widehat{\Phi}, \widehat{\eta})$ such that $\widehat{U} = O = U \times Y$, $\widehat{X} \in \mathcal{P}(X)$ and $\widehat{Y} = X$. □

Notice the input to this $\widehat{\mathcal{M}}$ finite machine correspond to an input/output pair associated with the $\mathcal{M}$ finite machine; and $\widehat{\mathcal{M}}$'s outputs correspond to $\mathcal{M}$'s states This output is intended to be an *estimate* of the state of the finite machine $\mathcal{M}$, given the observed input/output.

Definition 2.1.12 (Convergence) A classical dynamical observer system $\widehat{\mathcal{M}}$ is said to be *initial-state-convergent* in finite time, *denoted by* $\widehat{\mathcal{M}_I}$, if there exists $K \in \mathcal{Z}_+$, such that, for all $N > K$ and for all $o_1^N \in O^*$ there exists $x_1 \in X$ such that

$$\forall \widehat{x_1} \in \widehat{X},\ \hat{\eta}(\hat{\Phi}(\widehat{x_1}, \widehat{u_1^N})) = x_1 \tag{2.9}$$

where $\widehat{u_1^N} = o_1^N$ and if we denote $x_k = \Phi(x_1, u_1^{k-1})$ for $k = 1, 2, \cdots, N$ then the relation $CS(o_1^N, x_1^N)$ must hold.

Similarly, a classical dynamical observer system $\widehat{\mathcal{M}}$ is said to be *current-state-convergent* in finite time, denoted by $\widehat{\mathcal{M}_C}$ if there exists a $K \in \mathcal{Z}_+$, such that, for all $N > K$ and for all $o_1^N \in O^*$ there exists a $x_1 \in X$, such that

$$\forall \widehat{x_1} \in \widehat{X},\ \hat{\eta}(\hat{\Phi}(\widehat{x_1}, \widehat{u_1^N})) = \Phi(x_1, u_1^{N-1}) \tag{2.10}$$

where $\widehat{u_1^N} = o_1^N$ and if we denote $x_k = \Phi(x_1, u_1^{k-1})$ for $k = 1, 2, \cdots, N$ then the relation $CS(o_1^N, x_1^N)$ must hold. □

Notice this means that the current-state-convergent (initial-state-convergent) observer finite machine, $\widehat{\mathcal{M}_C}$ (or $\widehat{\mathcal{M}_I}$), can start from *any* state, and then observe the behavior of the finite machine $\mathcal{M}$, which, itself, can start in any state. Given enough observations, $\widehat{\mathcal{M}_C}$ (or $\widehat{\mathcal{M}_I}$) will be able to determine $\mathcal{M}$'s current (initial) state, and will then stay *locked on,i.e.*, always giving $\mathcal{M}$'s state as output, for any given subsequent (sequence of) input/output pairs.

For notation, we will often refer to a (classical) dynamical observer finite machine as a CDO, and the finite machine being observed as the *base machine*.

2.2 Constructing Initial and Current State Observers

This section first presents graphical representations of the observation sets in the forms of the tree structures. These trees–we shall later call them initial or current state observer trees–present a graphical way of constructing classical dynamical observers. Then we prove an important property of observable machines: Theorem 2.2.1 states that notion of *observability* of a finite machine and the existence of a *convergent classical dynamical observer* are equivalent.

2.2.1 Initial and Current State Observer Trees

In the previous section, we presented an algebraic representation of the observation sets in terms of a recursive set operations. The information contained in these algebraic expressions can also be represented graphically, in terms of initial and current state observer trees. Starting from an empty observation, corresponding to any moment before the base machine is initialized, Equations 2.5 or 2.7 gives $ISE([]) = X$. Thus X is the root of an initial (or a current) state observer tree. Now, at the initial state, the observer reads the first observation $o_1 = \langle \phi, y \rangle$, and therefore the observer is able to improve its estimated initial (or current) state estimates via Equation 2.5 or 2.7 to conclude:

$$ISE([\langle \phi, y \rangle]) = \widehat{\{x_1\}}(o_1) = X \bigcap \eta^{-1}(y)$$

(or by Equations 2.6 or 2.8 to conclude: $CSE([\langle \phi, y \rangle]) = \widehat{\{x_1\}}(o_1) = \eta^{-1}(y)$).

Based on the value of y we may have different $ISE's$ (or $CSE's$). This leads to a splitting from the root node into new subnodes. Carrying on these operations, we end up with an initial (or current) state observer tree $OT_I(\mathcal{M})$ (or $OT_C(\mathcal{M})$) for a given finite machine $\mathcal{M}$. In the following example, we show how to generate an initial and a current state observer tree for a given finite machine.

Example 2.2.1 Figure 2.5 and 2.4 show the initial and the current state *observer trees* for the finite machine $\mathcal{M}_7$ given in Figure 2.1.

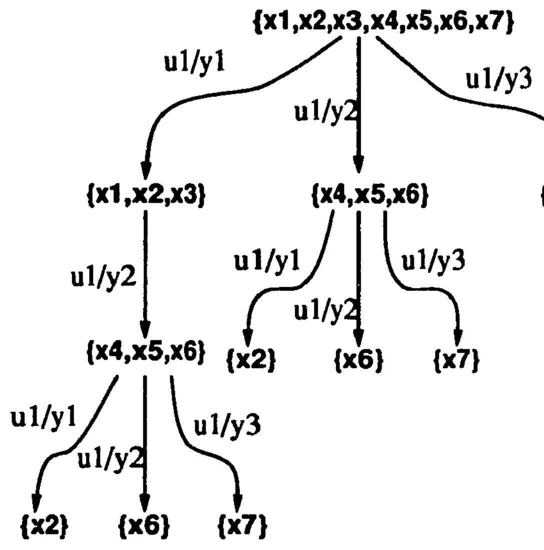


Figure 2.4: Current State Observer Tree for $\mathcal{M}_7$

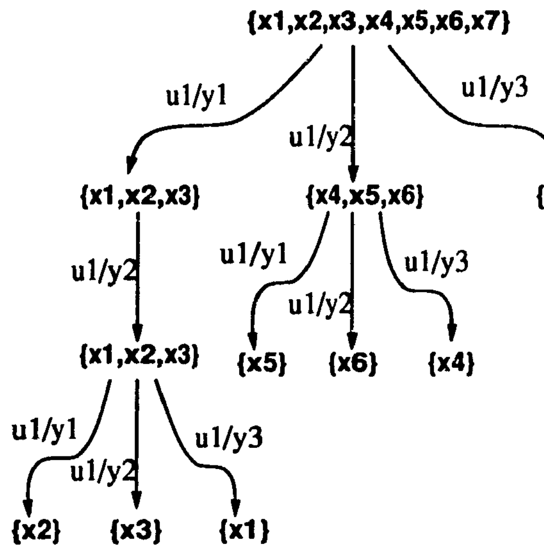


Figure 2.5: Initial State Observer Tree for $\mathcal{M}_7$

An observer tree will stop expanding at any singleton node since a singleton node can only have singleton nodes as its subnodes. □

We can view any observer machine, $\widehat{\mathcal{M}} = (\widehat{X}, \widehat{U}, \hat{Y}, \hat{\Phi}, \hat{\eta})$, as a directed, labeled graph, whose nodes correspond to the element set, $\widehat{X}$, and whose arcs correspond to the $\hat{\Phi}$ function: node $\widehat{x_1}$ is connected to $\widehat{x_2}$ by an arc labeled by $\hat{u}$ if and only if $\hat{\Phi}(\widehat{x_1}, \hat{u}) = \widehat{x_2}$.

An observer tree, i.e., a directed, labeled graph, will be called an *observer dag*, i.e., a directed, acyclic graph when the base machine is observable.

Obviously, the most general node is the root, and the singleton nodes are the leafs. To show that observer dags are true to their name, we need to show that they can have no cycles. The proof is obvious: a cycle in this (sub)graph necessarily renders the base machine non-observable.

Observer dags include exactly the traversals necessary to move from the most general node to the answer that is, to the correct set value of either the base machine's initial state, or its current state. This assumes that $\widehat{\mathcal{M}}$ begins in its most general node, and the data it receives is accurate (that is, it is observing the appropriate finite machine, and perceives its actual input and output). □

2.2.2 A System Is Observable Iff A Convergent CDO Exists

This subsection we prove that a finite machine is observable (initial or current state) if and only if there exists a convergent classical dynamical observer (initial or current state).

Theorem 2.2.1 (Equivalence) For any input-state-output finite machine, $\mathcal{M} = (X, U, Y, \Phi, \eta)$, the following statements are equivalent:

(i) $\mathcal{M}$ is initial (respectively current) state observable.

(ii) There exists a convergent $CDO(\mathcal{M})$ for the initial (respectively current) state value.

Proof

To show (i) implies (ii), it suffices to define an appropriate observer machine, $\widehat{\mathcal{M}_I}$ and $\widehat{\mathcal{M}_C}$ respectively, for any given base machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$, such that by

Definition 2.12 they are initial state convergent and current state convergent respectively.

This will be done by first constructing the initial and current state estimate dags, these correspond to the sets of initial and current state estimates and the transitions between them generated by the Equations 2.5 and 2.6 respectively. We then create the states and transitions of $\widehat{\mathcal{M}}_I$ and $\widehat{\mathcal{M}}_C$ by identifying (1) the state of $\widehat{\mathcal{M}}_I$ and $\widehat{\mathcal{M}}_C$ with the sets of state estimates in the appropriate dag, and (2) the transitions in $\widehat{\mathcal{M}}_I$ and $\widehat{\mathcal{M}}_C$ with the transitions in the appropriate dags.

To prove (ii) implies (i) we shall show that not (i) implies not (ii). Suppose $\mathcal{M}$ was such that it was not initial (respectively current) state observable. Then there would exist initial states $x, x' \in X$ such that $x \neq x'$ and for all N we would have $\widehat{\{x_1\}}(o_1^N) \supseteq \{x, x'\}$ $(\widehat{\{x_N\}}(o_1^N) \supseteq \{x_N = \Phi(x, u_1^{N-1}), x'_N = \Phi(x', u_1^{N-1})\}$ respectively), where $o_1^N = (u_1^{N-1}, y_1^N(u_1^{N-1}, x)) = (u_1^{N-1}, y_1^N(u_1^{N-1}, x'))$ for all $N \in \mathcal{Z}_+$.

But then no non-anticipative function and in particular no observer machine could map $o(x)_1^N$ into x_1 (x_N, respectively) in just those cases when x_1 is the initial state, (current, respectively) for all sufficient large N.

□

2.3 Complexity of the Observability Dags

This section provides theorems which describe the size of observer dags for finite machines bounding both their maximal depth, and their total number of nodes, for both initial and current state observer dags.

First let us present the following lemma which describes an interesting structural property of initial state observer trees.

Lemma 2.3.1 (Push-Down Property) For any $k \in \mathcal{Z}_+$ and for any $o_1^k \in O^*$ the following relation holds:

$$\widehat{\{x_1\}}(o_1^k) \subseteq \widehat{\{x_1\}}(o_1^{k-1}) \tag{2.11}$$

for any initial state estimate set $\widehat{\{x_1\}}(o_1^k)$ of any finite machine $\mathcal{M}$.

Proof

This is obvious from Equation 2.5 of Theorem 2.1.3, which says that an initial state estimate set at k is always contained in the initial state estimate set at $k-1$. □

This lemma simply states that any node in an initial state observer tree must be a subset of its parent node.

Theorem 2.3.1 ([CGW91] Size of Initial State Observability Dags – I) Let $\mathcal{M} = (X, Y, \Phi, \eta)$ be an *initial state observable* state-output finite machine with $|X|$ states, and let $\widehat{\mathcal{M}_I}$ be the initial state dag observer for $\mathcal{M}$. Then we have:

(i) $\widehat{\mathcal{M}_I}$ has depth at most $|X|$ before it converges to all singleton nodes.

(ii) $\widehat{\mathcal{M}_I}$ has at most $2|X|-1$ distinct nodes.

Proof

Proof of (i). Let S_t be the set of possible initial state estimates, at time t, *i.e.*, $S_t = \{\widehat{\{x_1\}}(y_1^t) | y_1^t \in Y^*\}$. Notice, of course that, $S_0 = \{X\}$, and $S_T = \{\{x\}\}$ if every element of X has been distinguished at time T; and $|S_t| \leq |X|$ for all t. Notice these S_ts include all and only the nodes in the observer dag, *i.e.*, the observer dag's nodes are exactly $\bigcup_t S_t$.

We can use S_t to define $\Pi_t \subset X \times X$ to be the set of pairs of states which are indistinguishable at time t: $\Pi_t(x_1, x_2) \Leftrightarrow \exists \Gamma \in S_t$, such that $x_1, x_2 \in \Gamma$. $S_0 = \{X\}$ means that $\Pi_0 = X \times X$, *i.e.*, before any observations, no pair is distinguishable.

We can define Π_t recursively:

$$\begin{aligned}\Pi_{t+1}(x,x') &\iff \bigwedge_{i=0}^{t}\eta(\Phi^i(x)) = \eta(\Phi^i(x'))\\ &\iff \eta(x)=\eta(x')\wedge\Pi_t(\Phi(x),\Phi(x'))\end{aligned}$$

The depth of the observer dag is the minimum t such that Π_t has converged. Notice that the Π_t sequence converges whenever $\Pi_{T+1} = \Pi_T$ It suffices to show that $\Pi_{T+2}(x,x') \Leftrightarrow \Pi_T(x,x')$; then by simple induction, we have $\Pi_{T+k} = \Pi_T$ for all $k \geq 0$

To show that $\Pi_{T+2}(x,x') \Leftrightarrow \Pi_T(x,x')$, observe:

$$\begin{aligned}\Pi_{T+2}(x,x') &\iff \eta(x)=\eta(x')\wedge\Pi_{T+1}(\Phi(x),\Phi(x'))\\ &\iff \eta(x)=\eta(x')\wedge\Pi_T(\Phi(x),\Phi(x'))\\ &\iff \Pi_{T+1}(x,x')\\ &\iff \Pi_T(x,x')\end{aligned}$$

Notice that $\Pi_{s+1} \neq \Pi_s$ means that $|S_{s+1}|$ is strictly greater than $|S_s|$, by the Lemma 2.3.1. As $|S_T| \leq |X|$, this strict increase can happen at most $|X|$ times, hence, we must have $T \leq |X|$ and hence the dag can have depth at most $|X|$.

Proof of (ii).

For any $\Gamma \subset 2^X$, let $\tau(\Gamma)$ be the observer dag rooted at Γ. (Hence $\tau(X)$ is the observer dag for the entire finite machine $\mathcal{M}$.) Let $|\tau(\Gamma)|$ represent the number of distinct nodes in any such tree.

We prove, by induction, that $|\tau(\Gamma)| \leq 2\times|\Gamma|-1$. This induction uses the $\widehat{\Phi}$-*order* of $|\Gamma|$, where the partial order $\widehat{\Phi}$-*order* is defined as

$$\widehat{\Phi}\text{-}order(\Gamma_1,\Gamma_2) \iff |\Gamma_1|>1 \text{ and } \exists y\ \widehat{\Phi}(\Gamma_1,y)=\Gamma_2$$

This $\widehat{\Phi_I}$ is the (initial state) observer machine's transition function, see Definition 2 2.1 As we are dealing with initial state observable finite machine and $\widehat{\Phi}$-*order* is false for all pairs of singleton sets and $\widehat{\Phi} - order$ is transitive, it follow that $\widehat{\Phi}$-*order* is a partial ordering.

Initialization of the induction: The minimal $\widehat{\Phi}$-*order* sets are singletons — *i.e.*, when $\Gamma = \{x\}$. Here, the observer tree, $\tau(\{x\})$, contains only the single node, $\{x\}$. Hence, $|\tau(\{x\})| = 1 \leq 2*1-1 = 2*|\Gamma|-1$.

General induction step: Take any non-minimal Γ; these correspond to non-singleton sets, $|\Gamma| > 1$. Using the superscript k to mean that the set Γ appeared at time k, we give the following diagrammatic representation for $\tau(\Gamma^k)$:

$$\begin{tikzcd}
& \Gamma^k \arrow[dl, dash, "y_1"'] \arrow[d, dash, "y_i"] \arrow[dr, dash, "y_m"] & \\
\tau(\Gamma_1^{k+1}) & \ldots\tau(\Gamma_i^{k+1})\ldots & \tau(\Gamma_m^{k+1})
\end{tikzcd}$$

where each $\Gamma_i^{k+1} = \{x \in \Gamma^k : \eta(\Phi^{k+1}(x)) = y_i\}$ and $y_i \in Y$. (Notice $\widehat{\Phi}$-*order* $(\Gamma^k, \Gamma_i^{k+1})$ holds for each i.) These Γ_i^{k+1}s are distinct (*i.e.*, $i \neq j \implies \Gamma_i^{k+1} \cap \Gamma_j^{k+1} = \phi$), and their union is a subset of Γ^k. As a final bit of notation, let $I = \{\, i \,|\, \Gamma_i^{k+1} \neq \{\} \,\}$.

There are three cases to consider:

Case 1: $|I| = 1$ and $\Gamma^k = \Gamma_i^{k+1}$ for some i:

Here, the $\tau(\Gamma^k)$ is of the form

$$\begin{tikzcd}
\Gamma^k \arrow[d, dash, "y_i"] \\
\tau(\Gamma_i^{k+1})
\end{tikzcd}$$

Using the fact that $|\tau(\Sigma)|$ is measuring the number of distinct nodes in the tree $\tau(\Sigma)$ rooted at the node Σ, and that $\Gamma^k = \Gamma_i^{k+1}$, we see[1] that $|\tau(\Gamma^k)| = |\tau(\Gamma_i^{k+1})|$. As

[1]Notice the finite machine may still be initial state observable(i s o.): Consider $X = \{x_0, \ldots x_5\}$ where

$\widehat{\Phi}$-*order*(Γ^k, Γ_i^{k+1}), we can inductively assume that $|\tau(\Gamma^k)| \leq 2 \times |\Gamma_i^{k+1}| - 1$, which allows us to prove

$$|\tau(\Gamma^k)| \; = \; |\tau(\Gamma_i^{k+1})| \; \leq \; 2 \times |\Gamma_i^{k+1}| - 1 \; = \; 2 \times |\Gamma^k| - 1$$

as desired.

Case 2: $|I| = 1$ *and* $\Gamma_i^{k+1} \subset \Gamma^k$ *for some* i *but* $\Gamma_i^{k+1} \neq \Gamma^k$*:*
As $\widehat{\Phi}$-*order*(Γ^k, Γ_i^{k+1}), by inductive assumption $|\tau(\Gamma_i^{k+1})| \leq 2 \times |\Gamma_i^{k+1}| - 1 < 2 \times |\Gamma^k| - 3$, as $|\Gamma_i^{k+1}| \leq |\Gamma^k| - 1$. Hence, $|\tau(\Gamma^k)| = 1 + |\tau(\Gamma_i^{k+1})| < 1 + 2 \times |\Gamma^k| - 3 < 2 \times |\Gamma^k| - 1$

Case 3: $|I| \geq 2$*:*
By inductive assumption, $|\tau(\Gamma_i^{k+1})| \leq 2 \times |\Gamma_i^{k+1}| - 1$ for all $i \in I$ (as $\widehat{\Phi}$-*order*(Γ^k, Γ_i^{k+1}))
Notice that $|\tau(\Gamma^k)| = 1 + \Sigma_{i \in I} |\tau(\Gamma_i^{k+1})|$ and $|\Gamma^k| \geq \Sigma_{i \in I} |\Gamma_i^{k+1}|$.

From above,

$$\begin{aligned}
|\tau(\Gamma^k)| &= 1 + \Sigma_{i \in I} |\tau(\Gamma_i^{k+1})| \\
&\leq 1 + \Sigma_{i \in I} (2 \times |\Gamma_i^{k+1}| - 1) \\
&\leq 1 + (2 \times \Sigma_{i \in I} |\Gamma_i^{k+1}|) - |I| \\
&\leq (2 \times |\Gamma^k|) + 1 - |I| \\
&\leq (2 \times |\Gamma^k|) - 1
\end{aligned}$$

as $|I| \geq 2$.

This proves the theorem. □

Remark: we notice that in an initial state observer dag, those identical nodes can only occur along an observation path in the consecutive layers Further a finite machine is an unsynchronized device and so those identical nodes can be merged into a single node This is explained in the following example.

$\Phi(x_i) = x_{(i+2) \bmod 6}$ and $\eta(x_0) = \eta(x_1) = A$, $\eta(x_2) = \eta(x_3) = B$, $\eta(x_4) = C$, $\eta(x_5) = D$ Notice this finite machine is i s o , and that, while x_4 and x_5 are indistinguishable at both time 1 and time 2, they do eventually become distinguished (at time 3)

Example 2.3.1 The merging of identical nodes in the initial state observer dag of Figure 2.5 is illustrated in Figure 2.6.

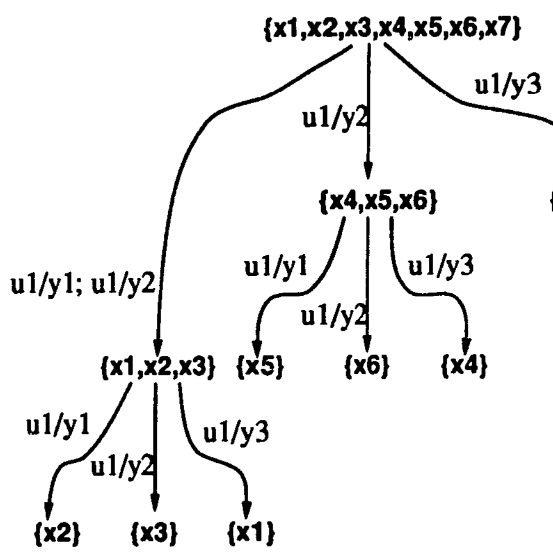


Figure 2.6: Merged Initial State Observer Dag for $\mathcal{M}_7$

By the above theorem there are at most $2|X|-1$ such nodes and hence we conclude an initial state observer can be built with at most $2|X|-1$ distinct states. □

Theorem 2.3.2 (Size of Initial State Observability Dags – II) Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be an *initial state observable* input-state-output finite machine with $|X|$ states, and let $\widehat{\mathcal{M}_I}$ be the initial state dag observer for $\mathcal{M}$. Then we have: $\widehat{\mathcal{M}_I}$ has depth at most $|X|$ before it converges to all singleton nodes.

Proof

Proof is similar to that of the part (i) of Theorem 2.3.1 plus the facts of that Lemma 2.3.1 and the merging of identical nodes stated above. □

For the size of an initial state observer, the worst case, can reach the limit of $2^{|\eta^{-1}(y_i)|}$. This worst case is achieved by assuming that $|U| = 2^{|\eta^{-1}(y_i)|}$. Example 2.3.2 shows the existence of such a class of initial state observable finite machines whose initial state observer tree can reach the upper bound of the size limit stated as above formula. For this class of initial state observable finite machines the corresponding initial state observer tree contains all possible subsets of $|\eta^{-1}(y_1)| = |X|/3$ as its nodes, i e., $2^{|X|/3}$ distinct nodes.

Example 2.3.2 Figure 2.7 shows the existence of a class of input-state-output finite machines whose initial state observer tree has $2^{|\eta^{-1}(y_1)|}$ distinct nodes and requires $2^{|\eta^{-1}(y_1)|} - |\eta^{-1}(y_1)| - 2$ controls.

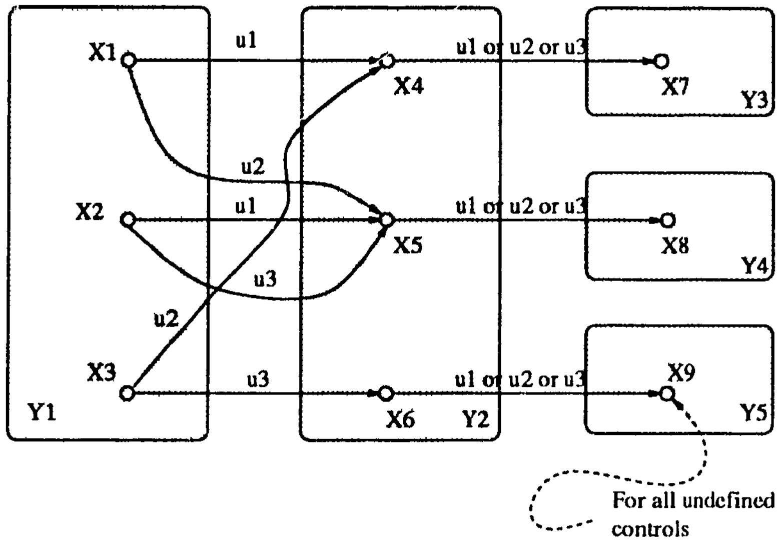


Figure 2.7: A nine state machine $\mathcal{M}_9$

The initial state observer tree for this nine state machine will reach its maximum number of nodes at the third layer which is $2^3 - 1 + 8$ distinct nodes. In general, one can construct initial state observable finite machines with the stated number of controls to split into all subsets of the preimage of $|\eta^{-1}(y_i)| = |X|/3$ for some y_i, i.e., $2^{|X|/3}$ distinct nodes. □

It has been noticed that the control set U plays an important roles in determining the size of the initial state observer dags The same is true for the current state observer dags. Hence the following size theorems will be stated in terms of current state observer dags for state-output and input-state-output finite machines respectively.

The following lemmas explain interesting features of current state estimate sets or the nodes in current state observer dags.

Lemma 2.3.2 (Push-Up Property) For any $k \in \mathcal{Z}_+$, for any $\widehat{\{x_{k+1}\}}(o_1^{k+1})$, there exists a current state estimate set $\widehat{\{x_k\}}(o_1'^k)$ such that:

$$\widehat{\{x_{k+1}\}}(o_1^{k+1}) \subseteq \widehat{\{x_k\}}(o_1'^k) \text{ if } o_{j+1} = o_j', \qquad \textit{for } j \in [1, \cdots, k] \tag{2.12}$$

□

Proof

The theorem can be proved by induction on k.

For $k = 1$, take any $o_1^2 = [\langle \phi, y_1 \rangle, \langle u_1, y_2 \rangle] = (o_1; o_2) \in O^2$ then

$$\begin{aligned} \widehat{\{x_2\}}(o_1^2) &= \Phi(\widehat{\{x_1\}}(o_1), u_1) \bigcap \eta^{-1}(y_2) \\ &= \Phi(\widehat{\{x_1\}}(o_1), u_1) \bigcap \widehat{\{x_1\}}(o_2) \\ &\subseteq \widehat{\{x_1\}}(o_2) \\ &= \widehat{\{x_1\}}(o_1') \end{aligned} \tag{2.13}$$

Since $\eta^{-1}(y_2) = \widehat{\{x_1\}}(o_2)$ and the condition: $o_1' = o_2$.

Let us make the induction hypothesis that for $k = n$ the required property is true.

Then for $k = n + 1$, take any $o_1^{n+1} = (o_1^n; o_{n+1}) \in O^{n+1}$, where $o_{n+1} = \langle u_n, y_{n+1} \rangle$ then

$$\begin{aligned} \widehat{\{x_{n+1}\}}(o_1^{n+1}) &= \Phi(\widehat{\{x_n\}}(o_1^n), u_n) \bigcap \eta^{-1}(y_{n+1}) && \text{(by Thm 2.1.3)} \\ &\subseteq \Phi(\widehat{\{x_{n-1}\}}(o_1'^{n-1}), u_n) \bigcap \eta^{-1}(y_{n+1}) && \text{(by hypothesis)} \\ &= \widehat{\{x_n\}}(o_1'^n) \end{aligned} \tag{2.14}$$

By taking the induction hypothesis and the conditions: $o'_{j-1} = o_j;\ \ for\ j \in [1, \cdots, n+1]$.

Hence the theorem holds for any $n \geq 1$. □

The above lemma simply states that any current state estimate $\widehat{\{x_{k+1}\}}(o_1^{k+1})$ with respect to o_1^{k+1} must be contained in some other current state estimate $\widehat{\{x_k\}}(o_1'^k)$ with respect to $o_1'^k$, where the observation sequence $o_1'^k$ equals to the observation sequence o_1^{k+1} with o_1 deleted.

The next lemma states that if any two nodes on any layers in the current state dag observer $\widehat{\mathcal{M}_C}$ for $\mathcal{M}$, are identical then all the subsequent nodes generated by these two nodes will be identical.

Lemma 2.3.3 (Same Sub-dag for Identical Nodes) Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a finite machine, and let $\widehat{\mathcal{M}_C}$ denote the current state dag observer for $\mathcal{M}$. Suppose for some $n > m \in \mathcal{Z}_+$ and some $o_1^m, o_1'^n \in O^*$ that

$$\widehat{\{x_m\}}(o_1^m) = \widehat{\{x_n\}}(o_1'^n) \tag{2 15}$$

then for all $o = \langle u, y \rangle \in O$

$$\widehat{\{x_{m+1}\}}(o_1^m; o) = \widehat{\{x_{n+1}\}}(o_1'^n; o) \tag{2.16}$$

□

Proof

From Theorem 2.1.3 we have the following equation.

$$\begin{aligned}
\widehat{\{x_{m+1}\}}(o_1^m; o) &= \Phi(\widehat{\{x_m\}}(o_1^m), u) \cap \eta^{-1}(y) && \text{(by Thm 2.1.3)} \\
&= \Phi(\widehat{\{x_n\}}(o_1'^n), u) \cap \eta^{-1}(y) && \text{(by hypothesis)} \qquad (2.17) \\
&= \widehat{\{x_{n+1}\}}(o_1'^n, o)
\end{aligned}$$

This proves the Lemma. □

The following corollary is an immediate consequence of the Lemma above. The corollary simply states that there are no two identical layers in a current state observer dag $\widehat{\mathcal{M}_C}$ for a current state observable finite machine $\mathcal{M}$.

Corollary 2.3.1 (No Identical Layers) Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a current state observable finite machine, and let $\widehat{\mathcal{M}_C}$ denote the current state dag observer for $\mathcal{M}$. Then we have for all $m, n \in \mathcal{Z}_+$, with $n \neq m$, and all $o_1^m \in O^m$ with $|\widehat{\{x_m\}}(o_1^m)| \neq 1$ then there exists $u_1^n \in O^n$ such that

$$\widehat{\{x_m\}}(o_1^m) \neq \widehat{\{x_n\}}(o_1'^n). \tag{2.18}$$

□

Proof

Suppose the contrary is true.

Then we have for some $m, n \in \mathcal{Z}_+$, with $n \neq m$, and for all o_1^m and $\widehat{\{x_m\}}(o_1^m)$ there exists a $\widehat{\{x_n\}}(o_1'^n)$ such that

$$\widehat{\{x_m\}}(o_1^m) = \widehat{\{x_n\}}(o_1'^n) \tag{2.19}$$

and for all $o_1'n$ and $\widehat{\{x_n\}}(o_1'^n)$, there exists a $\widehat{\{x_m\}}(o_1^m)$ such that

$$\widehat{\{x_n\}}(o_1'^n) = \widehat{\{x_m\}}(o_1^m) \tag{2.20}$$

Namely, the layer m and the layer n have exactly the same nodes.

By Lemma 2.3.2 just proved above, we know they will produce the same set of subnodes. Hence the $\widehat{\mathcal{M}_C}$ will never generate singleton nodes and this contradicts the acyclic property of $\widehat{\mathcal{M}_C}$ and hence the observability of $\mathcal{M}$.

The contradiction proves the corollary. □

The following theorem gives the size of the current state observer dags for *state-output* finite machines:

Theorem 2.3.3 ([CGW91] Size of Current State Observability Dags – I) Let $\mathcal{M} = (X, Y, \Phi, \eta)$ be a current state observable *state-output* finite machine and let $\widehat{\mathcal{M}_C}$ be the current state dag observer for $\mathcal{M}$. Then we have:

(i) $\widehat{\mathcal{M}_C}$ has depth at most $|X|$ before it converges to a layer of singleton nodes

(ii) $\widehat{\mathcal{M}_C}$ has at most $|X|^2$ nodes.

Proof

Proof of (i). We now let S_t refer to the set of possible current state estimates at time t, *i.e.*, $S_t = \{\widehat{\{x_t\}}(y_1^t) | y_1^t \in Y^t\}$. Once again, the nodes in the observer dag are exactly $\bigcup_t S_t$; furthermore, $|S_t| \leq N$ for all t, and $S_T = \{\{x\}\}$ if every element of X has been distinguished at time T.

As before, we let the indistinguishability class of sets $\sqcap_t \subset X \times X$ represent the set of pairs of states which are (current-state) indistinguishable at time t, *i.e.*, $\sqcap_t(x_1, x_2) \Leftrightarrow \exists \Gamma \in S_t,\ x_1 \in \Gamma\ \wedge\ x_2 \in \Gamma$. We can recursively define

$$\begin{aligned} \sqcap_1 &= ker(\eta) \\ \sqcap_2 &= ker(\eta) \cap \tilde{\Phi}(\sqcap_1) \\ &\ldots \\ \sqcap_{t+1} &= ker(\eta) \cap \tilde{\Phi}(\sqcap_t) \end{aligned}$$

where

$$ker(\eta) \overset{\text{def}}{=} \{< x_1, x_2 > \in X \times X \mid \eta(x_1) = \eta(x_2)\}$$

and

$$\tilde{\Phi}(\sqcap_t) \overset{\text{def}}{=} \{(x_1, x_2) \mid \exists x_1', x_2', \ni \sqcap_t(x_1', x_2') \wedge\ x_1 = \Phi(x_1') \wedge x_2 = \Phi(x_2')\}$$

We determine the maximum depth of this observer dag by finding the t such that $\sqcap_t$ has converged. As before, we have converged whenever $\sqcap_{T+1} = \sqcap_T$.

Suppose $\Pi_{T+1} = \Pi_T$ and consider Π_{T+2}:

$$\begin{aligned}\Pi_{T+2} &= ker(\eta) \cap \tilde{\Phi}(\Pi_{T+1}) \\ &= ker(\eta) \cap \tilde{\Phi}(\Pi_T) \\ &= \Pi_{T+1} \\ &= \Pi_T\end{aligned}$$

By simple induction, this means that $\Pi_{T+k} = \Pi_T \quad \forall k \geq 0$.

(Let T denote the time at which our current state observer $\widehat{\mathcal{M}_C}$ converges.)

By the Lemmas 2.3.2 and 2.3.3 and the Corollary 2.3.1 it suffices to show that $\Pi_{s+1} \neq \Pi_s$ means that $|S_{s+1}|$ is strictly greater than $|S_s|$. Then, as $|S_T| \leq |X|$, the depth of the dag can be no greater than $|X|$.

We need only prove that

$$\forall x_1, x_2 \in X, \ \Pi_{t+1}(x_1, x_2) \Rightarrow \Pi_t(x_1, x_2) \tag{2.21}$$

holds for all t. The proof is by induction:

The base case, $t = 0$, is trivial, as Π_0 is always true.

Assume Equation 2.21 holds for all $t \leq K$; we need only to show that for all $x_1, x_2 \in X$, $\Pi_{K+1}(x_1, x_2) \Rightarrow \Pi_K(x_1, x_2)$. Take any $\langle x_1, x_2 \rangle$ pair such that $\Pi_{K+1}(x_1, x_2)$ holds. By definition, this means

$$\eta(x_1) = \eta(x_2) \wedge \exists x_1', x_2', \Pi_K(x_1', x_2') \wedge x_1 = \Phi(x_1') \wedge x_2 = \Phi(x_2')$$

By inductive assumption, we know that $\Pi_{K-1}(x_1', x_2')$ must hold for this $\langle x_1', x_2' \rangle$ pair. Hence, $\Pi_{K-1}(x_1', x_2') \wedge x_1 = \Phi(x_1') \wedge x_2 = \Phi(x_2')$ holds, which is sufficient to guarantee that $\Pi_K(x_1, x_2)$ holds.

Proof of (ii). We know from (i) that this observer dag can have at most depth $|X|$. As the number of nodes at any given depth can be at most $|X|$, the total dag can have at most $|X|^2$ nodes. □

(This relationship is not necessarily true for current state observer dags of input-state-output finite machines since even singleton nodes can split under different inputs. In that case $|S_s| \leq |X|$ does not hold and this is the source of increased complexity for the case of input-state-output finite machines).

Next we present the theorem which states the size limit of current state observer dags for input-state-output finite machines.

Theorem 2.3.4 (Size of Current State Observability Dags – II) Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a current state observable *input-state-output* finite machine and let $\widehat{\mathcal{M}_C}$ be the current state dag observer for $\mathcal{M}$. Then we have: $\widehat{\mathcal{M}_C}$ has depth of $O(|X|^2)$ before it converges to a layer of singleton nodes.

Proof

We extend the current state indistinguishability relation, Π_k, defined in Theorem 2.3.3 to include the inputs as follows:

$$\begin{aligned}
\Pi_1 &= ker(\eta) \\
\Pi_2 &= ker(\eta) \cap \tilde{\Phi}(\Pi_1) \\
&\dots \\
\Pi_{t+1} &= ker(\eta) \cap \tilde{\Phi}(\Pi_t)
\end{aligned}$$

where

$$ker(\eta) \overset{\text{def}}{=} \{ < x_1, x_2 > \in X \times X \,|\, \eta(x_1) = \eta(x_2) \}$$

and

$$\tilde{\Phi}(\Pi_i) \overset{\text{def}}{=} \{ (x_1, x_2) \,|\, \exists x_1', x_2'. \Pi_i(x_1', x_2') \wedge x_1 = \Phi(x_1', u) \wedge x_2 = \Phi(x_2', u) \text{ for some } u \in U \}$$

Now by Lemma 2.3.3 we know within any single path of a current state observer dag there exist no identical nodes, since otherwise a cycle would exist and observability would be lost. Therefore the maximum number of possible indistinguishable pairs of states will be limited by $C_2^{|X|} = \frac{|X||X-1|}{2} \leq |X|^2$. This proves the limit on the maximum depth.

□

The size of a current state observer, in the worst case, can reach the limit of $2^{|\eta^{-1}(y_i)|}$. This worst case is achieved by assuming that $|U| = 2^{|\eta^{-1}(y_i)|}$. Example 2.3.3 shows the existence of such a class of current state observable finite machines whose current state observer tree can reach the upper bound of the size limit stated in the above formla. For this class of current state observable fin' machines $|\eta{-}1(y_1)| = |X|/2$ where the corresponding current state observer tree contains all possible subsets of $|\eta^{-1}(y_1)|$ as its nodes.

The case of that $|U|$ is bounded by a polynomial function of $|X|$ is still not clear. Example 2.3.3 shows how to construct such an input-state-output finite machines whose current state observer dags have $2^{|X|/2}$ number of nodes.

These bounds are tight as is shown by the following example.

Example 2.3.3 The depth of the current state dag observer for the class of finite machines achieves the upper bound $|X|^2$ due to the fact that the longest indistinguishable path will include all possible state pairs. Figure 2.8 is a 12 state machine which is current state observable and whose current state observer dag will have the maximum depth as specified by the result stated in part (i) of the above theorem. In this case the state output function η is defined by: $\eta(x_1) = \eta(x_2) = \cdots, = \eta(x_{10}) = y1$ and $\eta(x_{11}) = \eta(_{12}) = y2$.

The longest path in the current state observer dag for the twelve state machine traverses the elements above the main diagonal of the $X \times X$ matrix, in the following sequence of state pairs:

$$\{x_{11}, x_{12}\},$$

$$\{x_1, x_2\}, \{x_2, x_3\}, \cdots, \{x_9, x_{10}\},$$

$$\{x_1, x_3\}, \{x_2, x_4\}, \cdots, \{x_8, x_{10}\},$$

$$\{x_1, x_4\}, \{x_2, x_5\}, \cdots, \{x_7, x_{10}\},$$

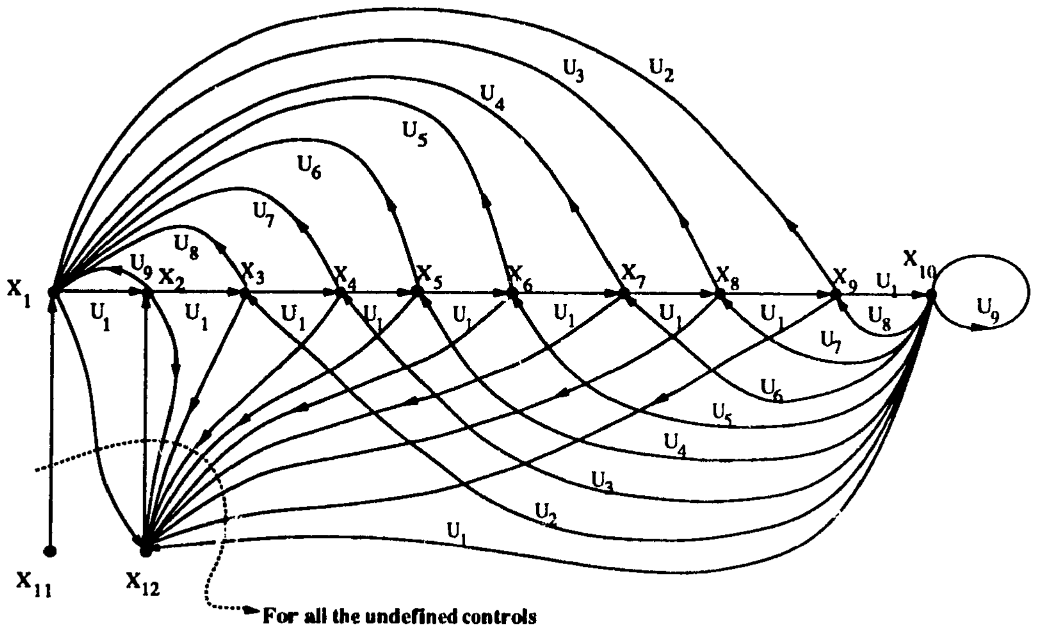


Figure 2.8: A twelve state machine

$$\{x_1, x_5\}, \{x_2, x_6\}, \cdots, \{x_6, x_{10}\},$$

$$\vdots$$

Due to the following controls:

$$u_1,$$

$$u_1, \cdots, u_1,$$

$$u_2, u_1, \cdots, u_1,$$

$$u_3, u_1, \cdots, u_1,$$

$$u_4, u_1, \cdots, u_1,$$

$$\vdots$$

This twelve state machine can be extended to a N state machine. The extension is by adding appropriate state elements to the y_1 group to the right of x_{10} and reassign the state transitions in a decreasing order of the $u_i's$ on top and reassign new controls to the left most (i.e., with highest index) state element in the y_1 group as illustrated by Figure 2.8.

An example of another class of current state observable finite machines which achieve the upper limit of the number of nodes in their current state observer dags is given in Figure 2.9. This gives a representative of a 6 state machine whose current state observer dag has the number of nodes in the range as specified.

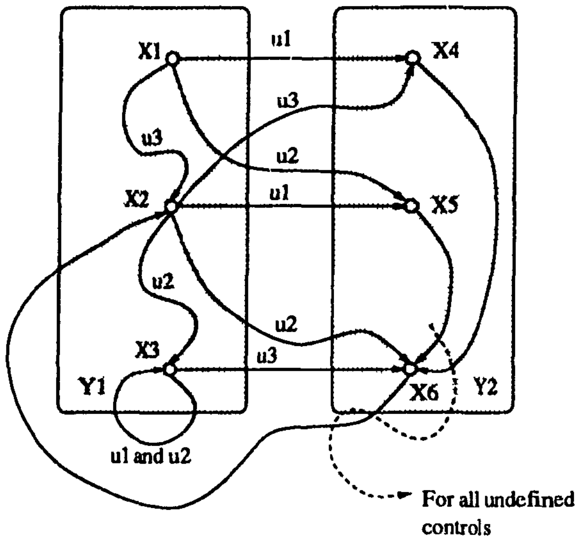


Figure 2.9: A six state machine

The current state observer dag for this six state machine will split into two nodes in the first layer since $|Y| = 2$. Then for $\eta = y1$ group, there are $2^{|\eta^{-1}(y1)|}$ subnodes in the second layer which reaches the upper limit. In general one can construct current state observable finite machines with the stated number of controls to split into all subsets of the preimage of $|\eta^{-1}(y_i)| = |X|/2$ for some y_i, i.e., $2^{|X|/2}$ distinct nodes. □

Next theorem connects the initial state estimate set with the current state estimate set by pushing forward the initial state estimate via the given observation sequence. The theorem shows that this forwarded image of an initial state estimate set is always contained in the corresponding current state estimate set.

Theorem 2.3.5 Given an observation sequence $o_1^N = \langle u_1^{N-1}, y_1^N \rangle \in O^N$ for a finite state machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$, the following relation holds:

$$\Phi(\widehat{\{x_1\}}(o_1^N), u_1^{N-1}) \subseteq \widehat{\{x_N\}}(o_1^N)$$

Proof

We prove this by induction over N as follows:

Case N=1:

$$\begin{aligned} LHS &= \Phi(\widehat{\{x_1\}}(o_1), u_1^0) = \widehat{\{x_1\}}(o_1) \\ &= \eta^{-1}(y_1) \\ &= \widehat{\{x_1\}}(o_1) \\ &= RHS. \end{aligned}$$

The following two facts are used in the inductive derivation of the result given below, let A and B be two sets and Φ a mapping, then

$$\Phi(A \bigcap B) \subseteq \Phi(A) \bigcap \Phi(B)$$

and

$$\Phi\Phi^{-1}(A) \subseteq A$$

The inductive proof then goes as follows:

Case N=2:

$$LHS = \Phi\Big(\bigcap_{k=1}^{2} \Phi^{-1}(\eta^{-1}(y_k), u_1^{k-1}), u_1\Big)$$

$$\begin{aligned}
&= \Phi\Big(\eta^{-1}(y_1)\bigcap\Phi^{-1}(\eta^{-1}(y_2),u_1),u_1\Big)\\
&\subseteq \Phi\Big(\eta^{-1}(y_1),u_1\Big)\bigcap\Phi\Big(\Phi^{-1}(\eta^{-1}(y_2),u_1),u_1\Big)\\
&\subseteq \Phi\Big(\eta^{-1}(y_1),u_1\Big)\bigcap\eta^{-1}(y_2)\\
&= \bigcap_{k=1}^{2}\Phi\Big(\eta^{-1}(y_k),u_k^{N-1}\Big)\\
&= RHS.
\end{aligned}$$

Induction hypothesis:

$$\Phi\Big(\bigcap_{k=1}^{n}\Phi^{-1}(\eta^{-1}(y_k),u_1^{k-1}),u_1^{n-1}\Big)\subseteq\bigcap_{k=1}^{n}\Phi\Big(\eta^{-1}(y_k),u_k^{n-1}\Big)$$

Case N=n+1

$$\begin{aligned}
LHS &= \Phi\Big(\bigcap_{k=1}^{n+1}(\Phi^{-1}(\eta^{-1}(y_k),u_1^{k-1}),u_1^{n}\Big)\\
&= \Phi\Big(\Phi^{-1}(\eta^{-1}(y_{n+1}),u_1^{n})\bigcap\bigcap_{k=1}^{n}\Phi^{-1}(\eta^{-1}(y_k),u_1^{k-1}),u_1^{n}\Big)\\
&\subseteq \Phi\Big(\Phi^{-1}(\eta^{-1}(y_{n+1}),u_1^{n}),u_1^{n}\Big)\bigcap\Phi\Big(\bigcap_{k=1}^{n}\Phi^{-1}(\eta^{-1}(y_k),u_1^{k-1}),u_1^{n}\Big)\\
&\subseteq \eta^{-1}(y_{n+1})\bigcap\Phi\Big(\widehat{\{x_n\}}(o_1^{n}),u_n\Big)\\
&= \widehat{\{x_{n+1}\}}(o_1^{n+1})
\end{aligned}$$

This proves the theorem. □

Furthermore, a sufficient condition is that when a finite machine is both initial state observable and that Φ is invertible for each state x then the subset relation becomes equality in Theorem 2.3.5.

Lemma 2.3.4 Given a finite machine $\mathcal{M}=(X,U,Y,\Phi,\eta)$, the following relation holds: $\Phi(A\bigcap B)=\Phi(A)\bigcap\Phi(B)$ for any $A,B\subseteq X$, if $\mathcal{M}$ is initial state observable.

Proof

We need only to show that $\Phi(A)\bigcap\Phi(B)\subseteq\Phi(A\bigcap B)$. Take any $x\in\Phi(A)\bigcap\Phi(B)$, there exist $x_A\in A$ and $x_B\in B$ such that $\Phi(x_A)=\Phi(x_B)=x$.

Now there are following cases to be considered:

Case 1. $x_A = x_B \in A \bigcap B$. Obviously $x \in \Phi(A \bigcap B)$ in this case.

Case 2. $x_A \neq x_B$ there are three more cases to consider here:

Case 2.1. $x_A \notin A \bigcap B$ but $x_B \in A \bigcap B$. Now Obviously we can take x_B and push it by Φ i.e., $x = \Phi(x_B) \in \Phi(A \bigcap B)$.

Case 2.2. $x_A \in A \bigcap B$ but $x_B \notin A \bigcap B$. In the same manner, we can take x_A and push it by Φ i.e., $x = \Phi(x_A) \in \Phi(A \bigcap B)$.

Case 2.3. $x_A \notin A \bigcap B$ and $x_B \notin A \bigcap B$. This will violate the initial state observability of $\mathcal{M}$ for state x_A and x_B, since under the same control, if two states are driven into a single state x then these two states will never be able to be separated again

This proves the lemma.

□

We conclude this chapter with the following theorem.

Theorem 2.3.6 Given an initial state observable finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ and Φ is invertible for each state $x \in X$, then for any observation sequence $o_1^N = \langle u_1^{N-1}, y_1^N \rangle \in O^N$ the following relation holds:

$$\Phi(\widehat{\{x_1\}}(o_1^N), u_1^{N-1}) = \widehat{\{x_N\}}(o_1^N)$$

Proof

Proof of this theorem is obtained from the proof of the Theorem 2 3 5 and Lemma 2.3.4 and the fact that $\Phi(x, u)$ is invertible for each state element $x \in X$. □

Chapter 3

Controllers for Finite Machines

The problem of steering the state of a partially observed finite machine (i.e., a partially observed deterministic finite input-state-output system) $\mathcal{M}$ to a desired terminal state is considered in this chapter. Using the framework described in the previous chapter we provide necessary and sufficient dynamic programming conditions for the controllability of $\mathcal{M}$. These conditions are stated in terms of backward recurrence equations involving the state estimate sets generated by classical dynamical current (or initial) state observers as we defined in Chapter 2. Early version of this work has been reported in [CW89a] as a joint work with Dr.P.E.Caines.

3.1 State Feedback Control

To steer a finite machine $\mathcal{M}$ from an initial state to a target state, using either complete or partial state observations, a standard close loop control approach, see Figure 3.1, is to build a controller, or a regulator, $Reg(\mathcal{M})$ which is fed with estimate of the state of $\mathcal{M}$, generated by a state observer, $Ob(\mathcal{M})$, and which then produces control inputs for $\mathcal{M}$.

The controller we discuss below will be constructed via a dynamic programming technique It turns out that $Reg(\mathcal{M})$ itself is representable by an input-state-output finite

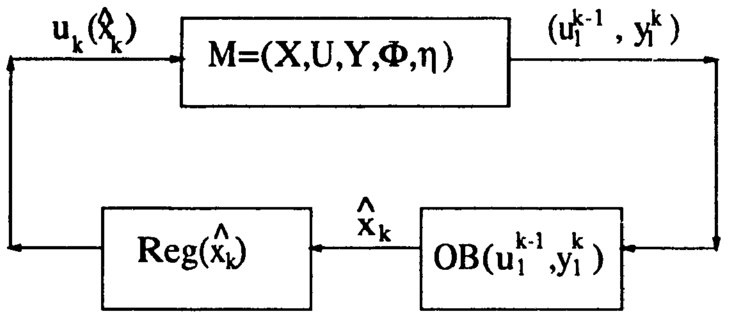


Figure 3.1: A closed loop controller

machine. Now let us start from basic definitions.

Definition 3.1.1 Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a finite machine and $x \in X$ be a state of $\mathcal{M}$. Then $R_k(x)$, $k \geq 1$ is said to be *the set of reachable states from x in less than or equal to k steps* if it satisfies the following relations:

$$\begin{aligned} R_0(x) &= \{x\}\overline{\nabla}\Phi^0(x,U) && \text{for } k=0 \\ R_1(x) &= R_0(x)\bigcup\Big\{x' : \exists u \in U, x' = \Phi(x,u)\Big\}\overline{\nabla}\Phi(x,U) && \text{for } k=1 \\ &\vdots \\ R_k(x) &= R_{k-1}(x)\bigcup\Phi(R_{k-1}(x),U)\overline{\nabla}\bigcup_{s=0}^{k}\Phi^s(x,U) && \text{for } k>1 \end{aligned}$$

□

We write $R(x) = \bigcup_{k=1}^{\infty} R_k(x)$ and the symbol R to denote the reachability relation for xRx' if and only if $x' \in R(x)$. In general R is transitive but not symmetric

Definition 3.1.2 A finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$, is said to be *controllable from* x if there exists $k = k(x) \geq 1$ such that $R_k(x) = X$ and is said to be *controllable* if $\mathcal{M}$ is controllable from x for every state $x \in X$. □

Theorem 3.1.1 A finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is controllable if and only if for all $x, x' \in X$, there exists $u_1^n \in U^*$, where n depends on x and x', and $n < |X|$ such that $\Phi(x, u_1^n) = x'$.

Proof

First we observe that

$$\begin{aligned} \mathcal{M} \text{ is controllable} &\iff \forall x \in X, \exists k = k(x) \geq 1 \; s.t. \; R_k(x) = X \\ &\iff \forall x, x' \in X, \exists u_1^n \in U^*, \\ &\qquad \text{where } n \leq \max_{x \in X}\{k(x)\} \text{ such that } \Phi(x, u_1^n) = x' \end{aligned}$$

So we need to show that if $n \geq |X|$ then there exist $u_1^{n'} \in U^*$ such that $n' < |X|$ and $\Phi(x, u_1^{n'}) = x'$. But this is clear, since if there exists u_1^n such that $\Phi(x, u_1^n) = x'$ and $n \geq |X|$ then there exist x_i and x_j in the state trajectory from x to x' induced by u_1^n such that $x_i = x_j$, i.e. there exists a loop in the state trajectory. Eliminate all such loops by choosing a new control sequence $u_1^{n'}$. The state trajectory from x to x' resulting from $u_1^{n'}$ is loop free and so $n' < |X|$ □

Corollary 3.1.1 If a finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is controllable, then for all $x \in X$, there exists a $k = k(x) < |X|$ such that $R_k(x) = X$ □

Proof

This follows directly from Definition 3.1.2 and Theorem 3.1.1 □

Theorem 3.1.2 Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a controllable finite machine. Then for all $x \in X$ and $k \geq 1$, the sequence of sets of reachable states from x, i.e., for all $k \in \mathcal{Z}_+, \{R_k(x) : k \geq 1\}$, is strictly monotonically increasing if $R_k(x) \neq X$, i.e., $R_k(x) \subset R_{k+1}(x)$ with $R_k(x) \neq R_{k+1}(x)$, if $R_k(x) \neq X$.

Proof

Assume this is not true, then we have some $x \in X$ and some $l \geq 1$ for which $R_l(x) \neq X$ and $R_l(x) = R_{l+1}(x)$. But this implies

$$\begin{aligned} R_{l+2}(x) &= \Phi(R_{l+1}(x), U)\bigcup R_l(x) \\ &= \Phi(R_l(x), U)\bigcup R_l(x) \\ &= R_{l+1}(x), \end{aligned}$$

and hence we have $R_{l+k}(x) = R_l(x) \subset X$ with $R_l(x) \neq X$ for all $k \geq 1$ which contradicts the controllability of $\mathcal{M}$. □

3.2 Construct State Feedback Controllers

In this section we describe how a state feedback controller for a finite machine (either a completely or partially observed finite machine) can be designed Our solution provided here is based on the dynamic programming principle.

3.2.1 Controllers for Completely Observed Finite Machines

Let us first examine the issue of how to control a completely observed finite machine, i e. a finite machine $\mathcal{M}$ for which η is $1-1$. To steer an initial state x into a target state x', one method for generating such a *state dependent control sequence, i e., a state dependent control law*, $u : X \longrightarrow U$, is to apply the dynamic programming technique (see for instance [Ber87, Cai88]). We may calculate a state dependent control law that steers the system between any two states by using a notion of the *k_{th} control pre-image of a target state.*

Definition 3.2.1 Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a competely observed finite machine, then the k_{th} *control pre-image* $R_c^{-k}(x^T)$ *of a target state* $x^T \in X$ is defined by the following recursive scheme:

$$\begin{aligned} R_c^0(x^T) &= \{x^T\} && \text{for } k=0 \\ R_c^{-k}(x^T) &= \bigcup_{u\in U}\Big\{x : \Phi(x,u) \in R_c^{-(k-1)}(x^T) \bigwedge x \notin R_c^{-s}(x^T) && \text{for } k \geq 1 \\ & \qquad\qquad \textit{for } 0 \leq s \leq k-1\Big\} \end{aligned}$$

□

Notice $R_c^{-k}(x^T)$ represents all the nodes which can be steered to home in *exactly* k steps.

Definition 3.2.2 A *set of good states at the stage* k, denoted by $G_c^k(x^T)$ *with respect to a target state* $x^T \in X$ of a completely observed finite machine $\mathcal{M}$ will be defined by $G_c^k(x^T) = \bigcup_{s=0}^{k} R_c^{-s}(x^T)$ □

That is a state x is a *good state* at the stage k with respect to x^T then x can be steered home in less than or equal to k steps.

Theorem 3.2.1 The completely observed finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is controllable from x if and only if for all $x^T \in X$, there exists $k = k(x^T, x) \leq |X|$ such that $x \in R_c^{-k}(x^T)$. Furthermore the following are equivalent:

(i) $\mathcal{M}$ *is controllable.*

(ii) *For each* $x^T \in X$ *there exists* $k = k(x^T) < |X|$ *such that* $G_c^k(x^T) = X$.

(iii) $G_c^l(x^T) \subset G_c^{l+1}(x^T)$ *with* $G_c^l(x^T) \neq G_c^{l+1}(x^T)$ *for* $l \geq 0$ *unless* $G_c^l(x^T) = X$.

Proof

Here we only prove (i) implies (iii) since the remaining implications are obvious.

$\mathcal{M}$ is controllable $\iff \forall x^T \in X, \exists k = k(x^T) < |X|$ such that $G_c^k(x^T) = X$.

Suppose $G_c^l(x^T) = G_c^{l+1}(x^T) \neq X$ for some l, then by the definition of $G_c^{l+1}(x^T) = \bigcup_{s=0}^{l+1} R_c^{-s}(x^T)$, we have

$$\begin{aligned} R_c^{-(l+1)}(x^T) &= \bigcup_{u \in U} \Big\{ x : \Phi(x,u) \in R_c^{-l}(x^T) \bigwedge x \notin R_c^{-s}(x^T) \\ &\qquad\qquad for\ 0 \leq s \leq l \Big\} \\ &= \phi. \end{aligned}$$

and hence $R_c^{-(l+j)}(x^T) = \phi$ for all $j \geq 1$, which yields a contradiction to (i) □

Theorem 3.2.2 (The Dynamic Programming Principle) Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a completely observed finite machine. $\mathcal{M}$ is controllable if and only if for any target state $x^T \in X$, there exists a control law $u_{x^T} : X \to U$ that steers any state $x \in X$ to the target state $x^T \in X$ of $\mathcal{M}$, i.e., there exists a control law which is only current state dependent.

Proof

We only prove $\Longrightarrow$ direction here since the other direction can be obtained easily

By Theorem 3.2.1, $\mathcal{M}$ controllable implies that for all $x^T \in X$, there exists $k < |X|$ such that $G_c^k(x^T) = X$. By the construction of the sequence of the control pre-images $R_c^{-s}(x^T)$ of the target state x^T, we see that a control law taking any given $x \in R_c^{-s}(x^T)$ to x^T in s steps is given by the sequence of controls u_1^s where u_1 takes any state $x \in R_c^{-s}(x^T)$ to some state $x' \in R_c^{-s+1}(x^T)$ and u_2 takes any state $x' \in R_c^{-s+1}(x^T)$ to some $x'' \in R_c^{-s+2}(x^T)$, and so on, until u_s takes any state in the resulting $R_c^{-1}(x^T)$ to x^T Now $G_c^k(x^T) = X$ for some k, and the sets $R_c^{-s}(x^T)$ are constructed recursively beginning only with the data x^T. Hence we see that there exists a control law, taking the state x to x^T whose values at any instant between $s = 1$ and $s = k$, depend only on the current state and the desired target state x^T. □

Example 3.2.1 Assume x_3 is the target state in the following completely observed, 7 state finite machine, called $\mathcal{M}_{7c}$

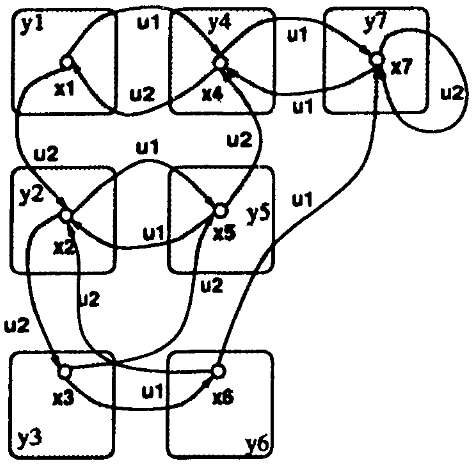


Figure 3 2: A completely observed finite machine $\mathcal{M}_{7c}$

The following is a control pre-image of the target state x_3

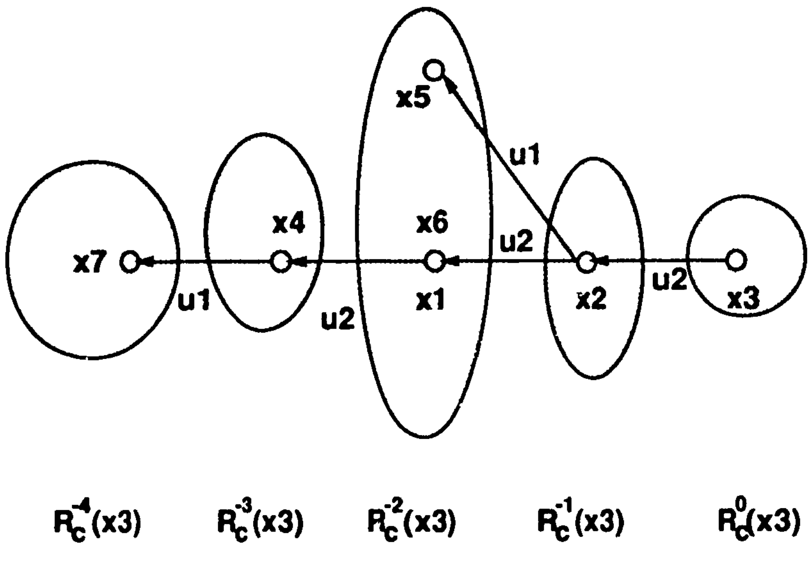


Figure 3.3: The control pre-image of the target state x_3 in $\mathcal{M}_{7c}$

Since $G_c^4(x_3) = X$ and then by checking each state for the controllability and the Theorem 3.2.2 we know that $\mathcal{M}_{7c}$ is controllable and the control law is defined by the above graph. □

Next we will turn to the more complex situation where the finite machine is partially observed.

3.2.2 Controllers for Partially Observed Finite Machines

Our first theorem on partially observed finite machines is as follows

Theorem 3.2.3 (Initial State Observable and Controllable – I) Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a partially observed initial state observable and controllable finite machine. Then for any $x, x^T \in X$ there exists $n \leq 2|X|$ and a control sequence $\mathrm{u}_1^n \in U^n$ such that $\Phi(x, \mathrm{u}_1^n) = x^T$

Proof

The above theorem simply states that for an initial state observable and controllable finite machine, any state can be steered into any other state in at most $2N$ steps, where $N = |X|$. The control sequence is constructed by first choosing arbitrary control inputs until the state estimate set converges to a singleton This will happen in less than or equal to N steps by Theorem 2.3.1 Then a control sequence is constructed according to Theorem 3 2 1 for a *completely observed finite machine* which will steer the singleton $\{\widehat{x_N}\}(o_1^N)$ to the target x^T in at most N steps. □

We have the following similar result for current state observable case

Theorem 3.2.4 (Current State Observable and Controllable – I) Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a partially observed current state observable and controllable finite machine. Then for any $x, x^T \in X$ there exists $n \leq |X|^2 + |X|$ and a control sequence $\mathrm{u}_1^n \in U^n$ such that $\Phi(x, \mathrm{u}_1^n) = x^T$

Proof

The above theorem simply states that for a current state observable and controllable finite machine, any state can be steered into any other state in at most $N^2 + N$ steps, where $N = |X|$. The control sequence is constructed by first choosing arbitrary control inputs until the state estimate set converges to a singleton. This will happen in less than or equal to N^2 steps by Theorem 2.3 4. Then a control sequence is constructed according to Theorem 3.2.1 for a *completely observed finite machine* which will steer the singleton $\{\widehat{x_N}\}(o_1^N)$ to the target x^T in at most $N^2 + N$ steps. □

A more challenging question is how to generate a control law which can be applied before the state estimate set has converged to a singleton.

We shall see that an answer to the question is obtained by applying the dynamic programming technique to the sequence of state estimate sets generated in the corresponding observer tree Let us first define a notion of a set of *good estimate states* with respect to a target state $x^T \in X$ among all the estimate states in a current state observer tree In the complete observation case, an estimated state can be specified simply as an element of the state set X; in the present case, however, we wish to indicate the position of a particular estimate state (i e., a subset of X) in the observer tree for $\mathcal{M}$. Since an estimate state (in particular, the singleton x^T) may appear at layers of the tree with a depth greater than any given value, we have to take a union over an infinite number of layers in the definition of the good estimate state sets.

Definition 3.2.3 An estimate state of a finite machine $\mathcal{M}$ is said to be a *good estimate state at the stage k* with respect to a given target state $x^T \in X$, if it satisfies the following recursive scheme:

1. Let $\{\widehat{x_N}\}(o_1^N)$ be an estimate state in the N-th layer of the current state observer tree for $\mathcal{M}$. If $\{\widehat{x_N}\}(o_1^N) = \{x^T\}$, then $\{\widehat{x_N}\}(o_1^N)$ is called a *good estimate state (at the stage 0)*. The set of all good estimate states at the stage 0, $H^0(x^T)$, is

the union of these estimate states, i.e.,

$$H^0(x^T) = \bigcup_{N=1}^{\infty}\left\{\widehat{\{x_N\}}(o_1^N) : \widehat{\{x_N\}}(o_1^N) = \{x^T\}\right\}$$

2. For $k \geq 1$, any estimate state $\widehat{\{x_N\}}(o_1^N)$ is a *good estimate state at the stage* k if there exists $\mathrm{u} \in U$, such that for all $y \in Y$, $\Phi(\widehat{\{x_N\}}(o_1^N), \mathrm{u}) \cap \eta^{-1}(y) = \widehat{\{x_{N+1}\}}(o_1^{N+1})$ and $\widehat{\{x_{N+1}\}}(o_1^{N+1})$ is itself a *good estimate state at the stage* $k-1$, or if $\widehat{\{x_N\}}(o_1^N)$ itself lies in $H^{k-1}(x^T)$, i.e.,

$$H^k(x^T) = H^{k-1}(x^T)\bigcup\Big\{\bigcup_{\mathrm{u}\in U}\bigcup_{N=1}^{\infty}\Big\{\widehat{\{x_N\}}(o_1^N) \,.\, \forall y \in Y$$
$$\Phi(\widehat{\{x_N\}}(o_1^N), \mathrm{u}) \cap \eta^{-1}(y) \in H^{k-1}(x^T)\Big\}\Big\}$$

□

In particular, we set $X \in H^k(x^T)$ for some k if for all $x \in X$, $\eta^{-1}(\eta(x)) \in H^k(x^T)$.

The following theorem states that an estimate state is good at the layer k of the observer tree if and only if it is good whenever it appears any where in the observer tree

Theorem 3.2.5 Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a finite machine $\widehat{\{x_l\}}(o_1^l) = \widehat{\{x_{l'}\}}(o_1^{l'})$ for any l and l', then $\widehat{\{x_l\}}(o_1^l) \in H^k(x^T)$, for some k, if and only if $\widehat{\{x_{l'}\}}(o_1^l) \in H^k(x^T)$

Proof

This is proved by the use of Lemma 2.3 3 and the Definition 3 2 3. □

Since state information is not directly obtainable, an observation based control law for a partially observed finite machine will be a function of observation sequences, which we shall write as $u_l : O^l \to U$. We will see that the notion of a good estimate state plays a key role in generating an observation based control law from the corresponding observer tree.

Next we give a definition of *controllability under the observation based output feedback controls.*

Definition 3.2.4 A partially observed finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is *partially observably controllable in less than or equal to N steps* if for all $x^T \in X$, there exists *an observation based feedback control law,* $u_l : O^l \to U$, for $N \geq l \geq 1$ such that for all $x_1 \in X$, there exists $k = k(x_1, x^T) \leq N$ and such that observation sequence $o_1^{k+1} = (u_1^k, y_1^{k+1}) \in O^{k+1}$, and $\bigcap_{j=1}^{k+1} \Phi(\eta^{-1}(y_j), u_j^k) = \{\widehat{x_{k+1}}\}(o_1^{k+1}) = \{x^T\}$. Since $x_1 \in \eta^{-1}(\eta(x_1))$, it follows immediately that $\Phi(x_1, u_1^k) = x^T$. □

Now we can prove that controllers are markovian with respect to state estimates.

Theorem 3.2.6 Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a partially observed finite machine. $\mathcal{M}$ is *partially observably controllable in less than or equal to N steps* if and only if for any fixed $x^T \in X$ there exists a $k \leq N$ such that $X \in H^k(x^T)$, i.e., for all $x_1 \in X$, there exists a $k = k(x_1) \leq N$ such that $\eta^{-1}(\eta(x_1)) \in H^k(x^T)$. Furthermore, the *observation based control law* can be expressed as a *state estimate based control law.*

Proof

Consider a fixed $x^T \in X$ and a fixed $N \in Z_+$.

$\Longrightarrow$

Suppose $\mathcal{M}$ is partially observably controllable in less than or equal to N steps. Then $\exists u_l : O^l \to U \; for \; N \geq l \geq 1$ such that $\forall x_1 \in X, \exists q = q(x_1) \leq N \; and \; o_1^{q+1} = (u_1^q, y_1^{q+1}) \in O^{q+1}$ such that $\bigcap_{j=1}^{q+1} \Phi(\eta^{-1}(y_j), u_j^q) = \{x^T\}$.

For any fixed x_1 apply the above argument to every $x \in \eta^{-1}(\eta(x_1))$, we have for any $x \in \eta^{-1}(\eta(x_1))$ there exists a $q = q(x) \leq N$ such that $\bigcap_{j=1}^{q+1} \Phi(\eta^{-1}(y_j), u_j^q) = \{x^T\}$. This implies that the estimate state set $\eta^{-1}(\eta(x_1))$ can be steered to x^T in less than or equal to $k = \max_{x \in \eta^{-1}(\eta(x_1))}(q(x)) \leq N$ Hence by Definition 3.2.3, $\eta^{-1}(\eta(x_1)) \in H^k(x^T)$.

$\Longleftarrow$

Suppose for all $x_1 \in X$ there exists a $k = k(x_1) \leq N$ such that $\eta^{-1}(\eta(x_1)) \in H^k(x^T)$. We need to show the existence of a control law $u_l : O^l \to U$ such that for all $x_1 \in X$ there exists a $q = q(x_1) \leq N$ such that $\bigcap_{j=1}^{q+1} \Phi(\eta^{-1}(\eta(x_1)), u_j^q(o_j^q)) = \{x^T\}$.

This can be proved by working forwards from $\eta^{-1}(\eta(x_1)) \in H^k(x^T)$ Assume for some $q \leq k \leq N$, $\eta^{-1}(\eta(x_1)) \in H^q(x^T)$. By the definition of $H^q(x^T)$, for some $\mathrm{u}_1 \in U$, $\Phi(\eta^{-1}(\eta(x_1)), \mathrm{u}_1) \cap \eta^{-1}(y) \in H^{q-1}(x^T)$. Set $u_1(o_1) = \mathrm{u}_1$, where $o_1 = \phi \times \eta(x_1)$. Then for all $y = \eta(\Phi(x, \mathrm{u}_1))$ as x varies over $\eta^{-1}(\eta(x_1))$, $\widehat{\{x_2\}}(o_1^2) = \Phi(\eta^{-1}(\eta(x_1)), \mathrm{u}_1) \cap \eta^{-1}(y)$ and $o_2 = \mathrm{u}_1 \times y$ satisfies $\widehat{\{x_2\}}(o_1^2) \in H^{q-1}(x^T)$ Hence it is clear there exists $u_2(o_1^2), u_3(o_1^3), \cdots, u_q(o_1^q)$ that steers $\widehat{\{x_2\}}(o_1^2)$ home. And so obviously there exists $u_1(o_1), u_2(o_1^2), \cdots, u_q(o_1^q)$ that steers $\eta^{-1}(\eta(x_1))$ to x^T, i e, $\bigcap_{j=1}^{q+1} \Phi(\eta^{-1}(y_j), \mathrm{u}_j^q) = \{x^T\}$ as required.

To prove the state estimate dependency of the controls we notice the fact that the controls are computed directly by use of the observer trees and as we move down through an observer tree, the state estimates are given by our basic recursive formulas of observation sets expressed in Equations 2.6 and 2.8

$$\begin{aligned}
\widehat{\{x_{N+1}\}}(o_1^{N+1}) &= \bigcap_{k=1}^{N+1} \Phi(\eta^{-1}(y_k), u_k^N) && (2.8)\\
&= \Phi(\widehat{\{x_N\}}(o_1^N), u_N) \bigcap \eta^{-1}(y_N) && (2.6)
\end{aligned}$$

The Equation 2.8 clearly states that the estimated state $\widehat{\{x_{N+1}\}}(o_1^{N+1})$ is a function of y_1^N and u_1^N and the Equation 2 6 simply says that $\widehat{\{x_N\}}(o_1^N)$ plays the same role as y_1^{N-1} and u_1^{N-1} do in Equation 2.8. This proves the stated property □

Theorem 3.2.7 (Initial State Observable and Controllable – II) If $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is *initial state observable and controllable,* then there exists a $n \leq 2|X|$ such that for all $x_1, x^T \in X$, $\eta^{-1}(\eta(x_1)) \in H^n(x^T)$, and hence $\mathcal{M}$ is partially observably controllable in less than n steps.

Proof

The initial state observable implies that any state estimate will converge to a singleton after at most $|X|$ steps by Theorem 2.3.1. Controllability implies that any target state can be reached from any other state in less than $|X|$ steps as stated in Theorem 3.2.1. Hence the theorem is established. □

We have the following similar result for the current state observable case.

Theorem 3.2.8 (Current State Observable and Controllable Systems– II) If $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is *current state observable and controllable,* then there exists a $n \leq |X|^2 + |X|$ such that for all $x_1, x^T \in X, \eta^{-1}(\eta(x_1)) \in H^n(x^T)$, and hence $\mathcal{M}$ is partially observably controllable in less than n steps.

Proof

The current state observable implies that any state estimate will converge to a singleton after at most $|X|^2$ steps by Theorem 2.3.4. Controllability implies that any target state can be reached from any other state in less than $|X|$ steps as stated in Theorem 3.2.1. Hence the theorem is established. □

Definition 3.2.5 Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a partially observed finite machine, $(\widehat{\mathcal{M}})_1^N$ be the current state observer dag for $\mathcal{M}$ with only the nodes in the first N layers. A *finite collection* $G^N(x^T)$ of a set of good estimate states with respect to a target state $x^T \in X$ is defined to be :

$$G^N(x^T) = H^N(x^T) \bigcap (\widehat{\mathcal{M}})_1^N$$

□

This $G^N(x^T)$ simply gives rise to the set of *good sets* in the first N layers of the current state observer dag of $\mathcal{M}$.

Theorem 3.2.9 A partially observed finite machine M is partially observably controllable in less than or equal to N steps if and only if for all $x^T \in X$, $X \in G^N(x^T)$.

Notice the use of the notion of a finite collection of a set of good estimate states in this theorem will give rise to an implementable criterion for testing the partially observably controllability in less than or equal to N steps for a partially observed finite machine. Furthermore, we can see that when the output function η in a given finite machine becomes one to one, then *a good estimate state* will be simply a *good state*, i e , $G^k(x^T) = G_c^k(x^T)$.

Example 3.2.2 We take the same finite machine as in Figure 3.2 but without the assumption that it is completely observed. The output function is given in the graph below, see Figure 3.4. It is a partially observed finite machine We can see this finite machine is observable and controllable in less than 7 steps via the observation tree based output(or estimated state) feedback controls.

To support this claim, next we first generate a part of the current state observer tree, in Figure 3.5 for this finite machine given in Figure 3.4 Assume our target state is x_3 again. Then we have the control pre-image or a collection of good sets with respect to the target state x_3 as shown in the Figure 3.6

□

The network generated by the dynamic programming technique, i e here we call it as a collection of good sets with respect to the target state, defines the control functions These control functions can also be given in the form of a finite machine which can be transformed from the network. We may call such a (finite) machine as a *regulator finite machine.*

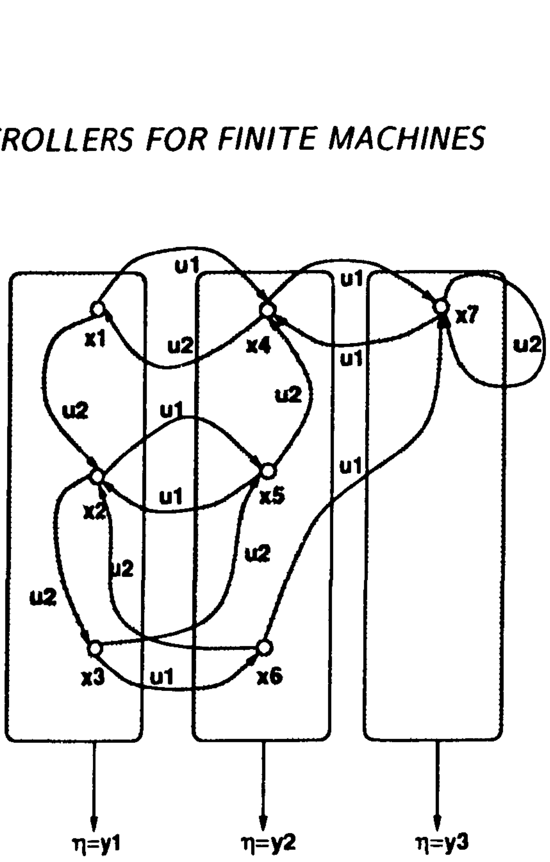


Figure 3.4: A partially observed finite machine $\mathcal{M}_{7p}$

3.3 Complexity of Classical Controller Design

In this section we address the complexity problem of designing classical controllers based on the algorithms presented in the previous section.

3.3.1 Complexity of Controller Design for Completely Observed Finite Machines

As we mentioned before, a completely observed finite machine is an input-state-output finite machine for which the output function η is taken to be one-to-one. So the controller has complete observation on the finite machine's state. The design of a controller based on the dynamic programming technique for such a finite machine is characterized by the notion of a set of good states with respect to a given target state as described in Definition 3.2.2 and the Theorems 3 2.1 and 3.2.2. In the following we analyze the complexity issue of the design of such a classical controller.

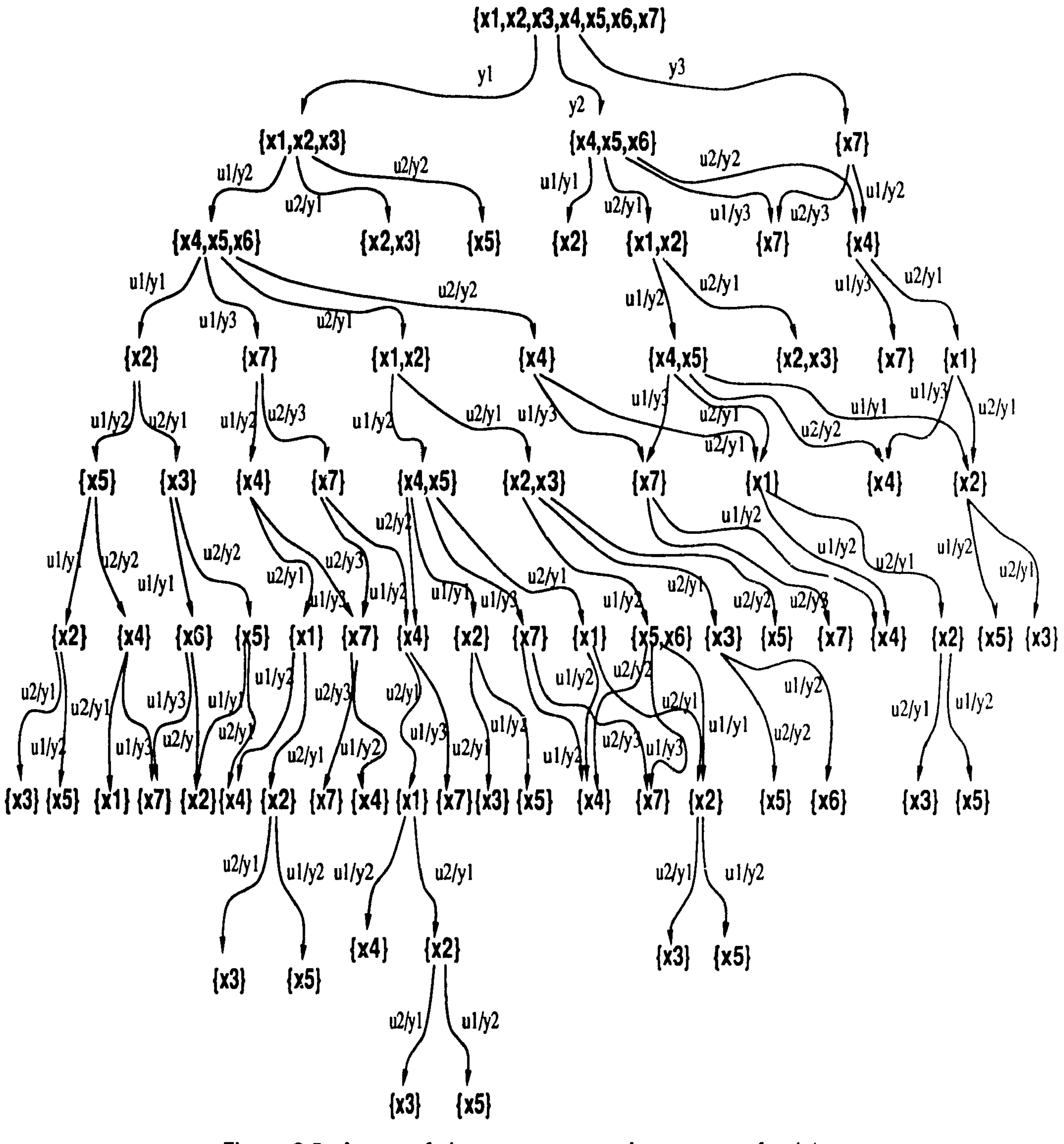


Figure 3.5: A part of the current state observer tree for $\mathcal{M}_{7p}$

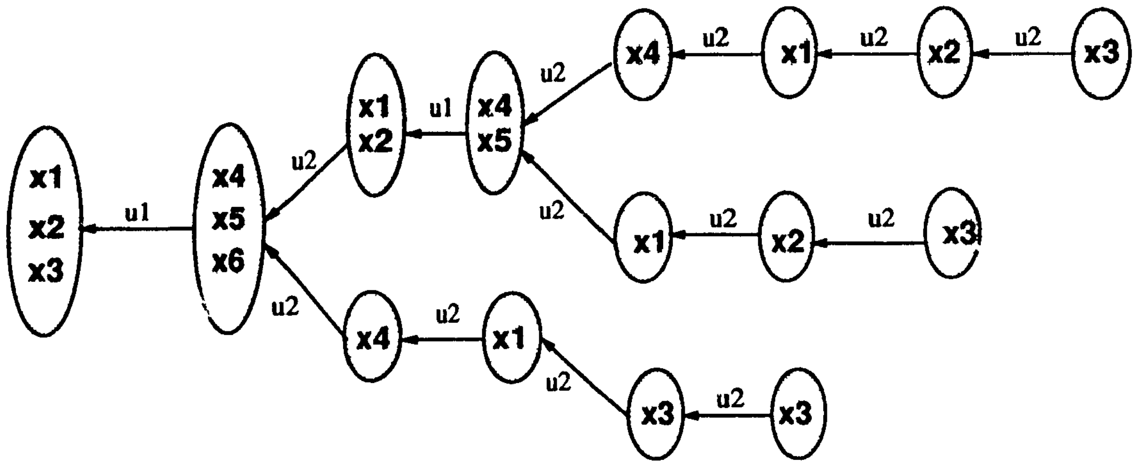


Figure 3.6: A part of the collection of the good sets w.r.t. the target state x_3 in $\mathcal{M}_{7p}$

We first define a *cost* function $c(x,x')$ on the state pairs by

$$c(x,x') = \begin{cases} 0 & \text{if } x = x' \text{ or} \\ 1 & \text{if } \exists u \in U \text{ s.t. } \Phi(x,u) = x' \\ \infty & \text{otherwise} \end{cases} \tag{3.1}$$

$c(\cdot,\cdot)$ can be thought of as an one step distance between any two states. Then the distance $d(\cdot,\cdot,\cdot)$ between any two states x, x' along a path $x^{(0)} = x, x^{(1)}, \cdots, x^{(k+1)} = x'$ generated by the control sequence u_1^k is defined by

$$d(x,x',u_1^k) = \begin{cases} 0 & \text{if } x = x' \\ \sum_{i=0}^{k} c(x^{(i)}, x^{(i+1)}) & \text{if } x^{(i+1)} = \Phi(x^{(i)}, u_i) \text{ with } x^{(0)} = x \\ & \text{and } x^{(k+1)} = x' \text{ for } u_1^k \in U^k \\ \infty & \text{otherwise} \end{cases} \tag{3.2}$$

Clearly the reachability relation given in Definition 3.1.1 can be redescribed in terms of $d(\cdot,\cdot,\cdot)$ via

$$\left\{x' \in \bigcup_{k=1}^{\infty} R_k(x)\right\} \quad \text{iff} \quad xRx' \tag{3.3}$$

$$\text{iff} \quad \exists u_1^k \in U^k, \text{ such that } d(x,x',u_1^k) < \infty \tag{3.4}$$

We shall define the minimum distance $d(x, x')$ between x and x' via

$$d(x, x') = \min_{k \in Z_1} \min_{u_1^k \in U^k} \left\{ d(x, x', u_1^k) \right\} \tag{3.5}$$

Evidently, x' is reachable from x if it is reachable in less than $|X| = N$ steps Hence the minimization in Equation 3.5 is over finite time (N) and a finite number of controls $|U|^N$.

An algorithm for designing a classical controller to steer x_s to x_T is given by the dynamic programming technique for the solution of Equation 3.5 Specifically we solve this by a backward iteration of

$$d_{j+1}(x', x_T) = \min_{x'' \in X} \left\{ c(x', x'') + d_j(x'', x_T) \right\} \tag{3.6}$$

with $d_0(\cdot, \cdot) = c(\cdot, \cdot)$ and $d(x_s, x_T) = d_j(x_s, x_T)$ for the smallest value of j such that $d_j(x_s, x_T) < \infty$. The resulting state-dependent control law is given implicitly as the sequence of controls that generate a sequence of states yielding the value $d(x_s, x_T)$

In considering the computational complexity of the evaluation of Equation 3 5 for all source and target states we shall specifying the following costs of elementary computational steps.

1. An evaluation of $\Phi(x, u)$ shall be counted as costing 1 unit of computation
2. The verification of the inclusion $x \in S \subseteq X$ shall count for $|S|$ units of computation.
3. The merging $A \cup B$ or the intersection $A \cap B$ of two subsets $A, B \subseteq X$ counts for $\max\{|A|, |B|\}$ units of computation.

Theorem 3.3.1 The computational complexity of the algorithm given by Equation 3.5, evaluated for all $x_s, x_T \in X$, is $O(|U| \times |X|^4)$.

Proof

First, we fix x_T and calculate $d(x_s, x_T)$ for each $x_s \in X$. We assume X be indexed by $X = \{x_1, x_2, \cdots, x_N\}$. The following staged diagram illustrates the argument.

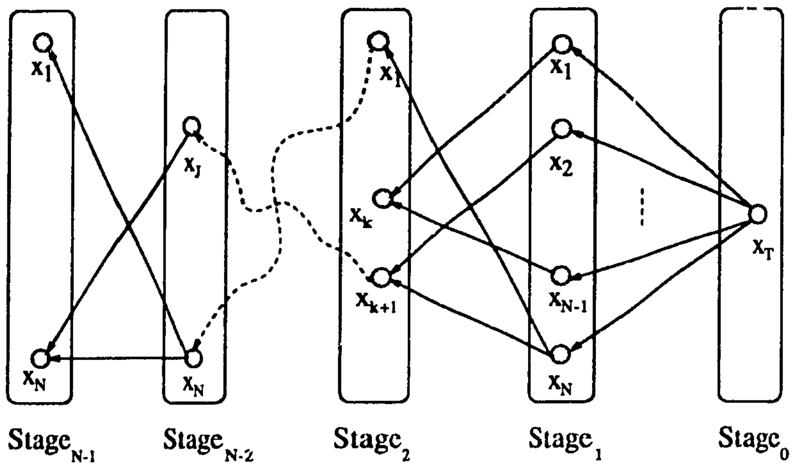


Figure 3.7: Staged Diagram

We assume $N = |X|$ in the following calculation. At stage 0, we need to make the following calculation· compute the cost of a transition from each of the $N-1$ nodes at stage 1 (excluding x_T itself) to x_T. This requires the evaluation of $\Phi(\cdot,\cdot)$ over $(N-1) \times |U|$ times, see Equation 3 2. Using the elementary computational cost table listed above, this costs $(N-1) \times |U|$ units of computation.

At stage $k+1$, for each target node $x'' \neq x_T$ we need to calculate the costs for $N-2$ transitions from $x' \notin \{x'', x_T\}$ to x''. Therefore the computation required for stage $k+1$ is $(N-2)(N-1) \times |U|$ units. By Theorem 3.1.1, if two states $x, x' \in X$ satisfy xRx', then $d(x, x') < N$. Hence we need to consider the calculation up to at most stage $N-1$. So for each fixed $x_T \in X$ we need to perform $[(N-2)^2(N-1)+(N-1)] \times |U|$ unit of calculations. Therefore the complexity of designing a classical controller via the dynamic programming Equation 3.5 is bounded by $O(|U| \times N[(N-2)^2(N-1)+(N-1)]) = O(|U| \times N^4)$. □

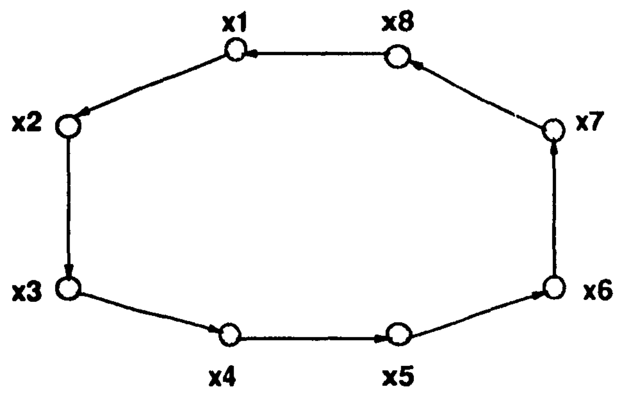


Figure 3.8: A Cyclic Finite Machine

Example 3.3.1 Consider a cyclic finite machine given in Figure 3 8

Take $x_5 \in X$ as the target state, then the staged diagram is given in Figure 3 9

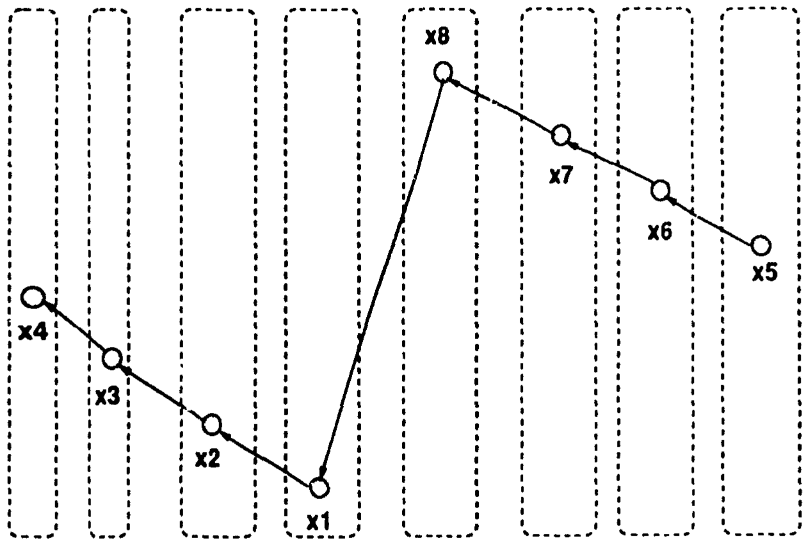


Figure 3.9: The Staged Diagram

and the computational cost for x_5 is: $(7 + 6^2 \times 7) \times 1$. □

The following calculation shows that the *good set* iteration of the Definition 3.2 2 has the same complexity as the dynamic programming calculation just described

First we rewrite the formula given in the Definition 3.2.2:

$$\begin{aligned} G_0(x^T) &= \{x^T\} \\ G_k(x^T) &= G_{k-1}(x^T)\bigcup\Big\{x \in X - G_{k-1}(x^T) : \exists u \in U, \\ &\qquad\qquad \Phi(x,u) \in G_{k-1}(x^T)\Big\} \qquad k \geq 1 \end{aligned}$$

Consider the computational cost required for the above recursive iterations, the cost for the merging of two sets can be ignored compare to the cost of creating new elements. Assume

$$\begin{aligned} n_0 &= |G_0(x^T)| \\ n_1 &= |G_1(x^T)| - |G_0(x^T)| \\ &\vdots \\ n_j &= |G_j(x^T)| - |G_{j-1}(x^T)| \\ &\vdots \end{aligned}$$

Where $\sum_{j=0}^{k} n_j = |X| = N$ and $k < N$. The units of computation required for calculating each $G_j(x^T)$ are listed below:

$$\begin{aligned} & 0 && j = 0 \\ & (N - n_0) \times |U| \times n_0 && j = 1 \\ & (N - (n_0 + n_1)) \times |U| \times (n_0 + n_1) && j = 2 \\ & \vdots \\ & (N - \sum_{i=0}^{k-1} n_i) \times |U| \times \sum_{i=0}^{k-1} n_i && j = k \end{aligned}$$

To sum up the above, we have for each $x^T \in X$:

$$|U| \times \sum_{j=1}^{k}\Big[N \times \sum_{i=0}^{j-1} n_i - \Big(\sum_{i=0}^{j-1} n_i\Big)^2\Big] \tag{3.7}$$

Therefore the worst case computational complexity for all $x^T \in X$ is obtained by taking the max of the above formula over $k < N$ and n_i where $i = 0, 1, 2, \cdots, k$ and $\sum_{i=0}^{k} n_i = |N|$, i.e.

$$\mathcal{C} = max_{k<N} max_{n_j, j=0,\cdots,k} \Big\{ |U| \times N \times \sum_{j=1}^{k} \Big[N \times \sum_{i=0}^{j-1} n_i - \Big(\sum_{i=0}^{j-1} n_i\Big)^2 \Big] \Big\} \qquad (3.8)$$

Evidently, by ignoring the negative part in the Equation 3.8 and taking note of $\sum_{i=0}^{j-1} n_i \leq N$ for any $j < k$, we get an upper bound on Equation 3.8, i.e.

$$\mathcal{C} \leq |U| \times N^4 = O(|U| \times N^4)$$

On the other hand, we can find a lower bound for this formula by taking a specific instance of Equation 3.8, i.e. taking $n_i = 1$ for $i = 0, 1, \cdots, k$, we get

$$\begin{aligned} \mathcal{C} \geq |U| \times N \sum_{j=1}^{N-1} (N \times j - j^2) &= |U| \times N (N \sum_{j=1}^{N-1} j - \sum_{j=1}^{N-1} j^2) \\ &= O(|U| \times N^4) \end{aligned}$$

Hence we have $\mathcal{C} = O(|U| \times N^4)$. □

3.3.2 Complexity of Controller Design for Partially Observed Finite Machines

In the case of partially observed finite machines, we use the *good estimate state* iteration algorithm based on the Definitions 3.2.3 and 3.2.5 given in Section 3.2.2. Here we will first rewrite the algorithm and then the analysis of its computational complexity

Assume $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is partially observably controllable in less than N steps The calculation of good estimate states at the stage k with respect to a target state $x^T \in X$ is as follows:

$$\begin{aligned}
G_0^N(x^T) &= \left\{ \widehat{\{x_N\}}(o_1^N) : \widehat{\{x_N\}}(o_1^N) = \{x^T\} \right\} && k = 0 \\
G_k^N(x^T) &= \Big\{ \widehat{\{x_{N-k}\}}(o_1^{N-k}) : \exists u \in U, \forall y \in Y, \ s.t. \\
&\quad \Phi(\widehat{\{x_{N-k}\}}(o_1^{N-k}), u) \bigcap \eta^{-1}(y) \in G_{k-1}^N(x^T) \\
&\quad or \ \widehat{\{x_{N-k}\}}(o_1^{N-k}) = \{x^T\} \Big\} && 1 \le k < N
\end{aligned}$$

We assume the $|\widehat{X}|$ is the width of the observer tree in the following calculation. The units of computation required for calculating each $G_k^N(x^T)$ for a fixed $x^T \in X$ are listed below:

$$\begin{aligned}
&O(|\widehat{X}|) && k = 0 \\
&|U| \times |Y| \times O(|\widehat{X}|^3) && 1 \le k < N
\end{aligned}$$

By Theorem 3.2.7, an initial state observable and controllable finite machine is partially observably controllable in less than $N \le 2|X|$ steps. Hence the complexity of designing a classical controller for such a partially observably controllable finite machine will be $O(|U| \times |Y| \times |\widehat{X}|^3 |X|^2)$.

Part II

A Logic Based Control Theory for Finite Machines:COCOLOG—A Conditional Observer and Controller Logic

In part I, we defined observer and controller problems for finite machines and showed how to construct observers and controllers directly in terms of finite machines. All of this discussion was in terms of the classical functional (or procedural) approach of building dynamical systems In this part of the thesis, we show a logic based (or declarative) approach can also be adopted to the same control tasks by introducing a new control theoretic paradigm, i.e., a framework of a *logic-based dynamical system* (LDS) which generates a sequence of propositions that correctly describe properties of the state of the given finite machine In particular, we are interested in those cases where the classical dynamical observer system estimates converge to the correct values of the systems state and the logic-based dynamical system *statements* converge (in an appropriate sense) to true characterizations of the system state. When such convergent observer systems exist we shall call the base finite machine *observable* or *logically observable* and similarly for *controllable* or *logically controllable* respectively.

To be more concrete, in order for a logic based system to be an embedded real time observer and controller we introduce a certain sequence of conventional logical systems. This sequence of structures constitute our main contribution of this thesis: COCOLOG – a conditional observer and controller logic. This is a precisely formulated control theoretic logic system, in which it is possible to express a large range of control problems and to find their solution by Automated Theorem Proving (ATP) method.

Chapter 4

Logic-Based Dynamical Systems

Previous two chapters presented the state estimation and control problems for system modeled by finite machines, in terms of classical functional (procedural) approach Here we introduce a logic-based framework for modeling and control of the same control problems

4.1 Logic-based System Modeling

Stated informally, a *logic* is a formalism capable of representing and reasoning about a world or worlds. It is usually viewed as a *language* — that is a set of *well formed formulas (or wffs)* and a set of *worlds* or *models* with respect to which the formulas are interpreted and a *truth function*, which assigns a truth value (usually true, false or unknown) to each wff.

The language together with its formal grammar is referred to as the *syntax* of the logic and the set of models and truth function as its *semantics*. A useful logic is able to encode the truth assignments of a large (possibly infinite) number of wffs in relatively few symbols, by employing a *derivation* or *inference process* In general, a derivation process is a symbolic manipulation process which maps one set of wffs into another set For notation, we write $\Sigma \vdash_{\alpha} \sigma$ to mean that the wff σ is derivable when applying the derivation process $\vdash_\alpha$ to the set of wffs Σ In this case the σ is called a *theorem* of Σ and the derivation is called a *proof.* Standard introductions to mathematical logic are [Men64, End72].

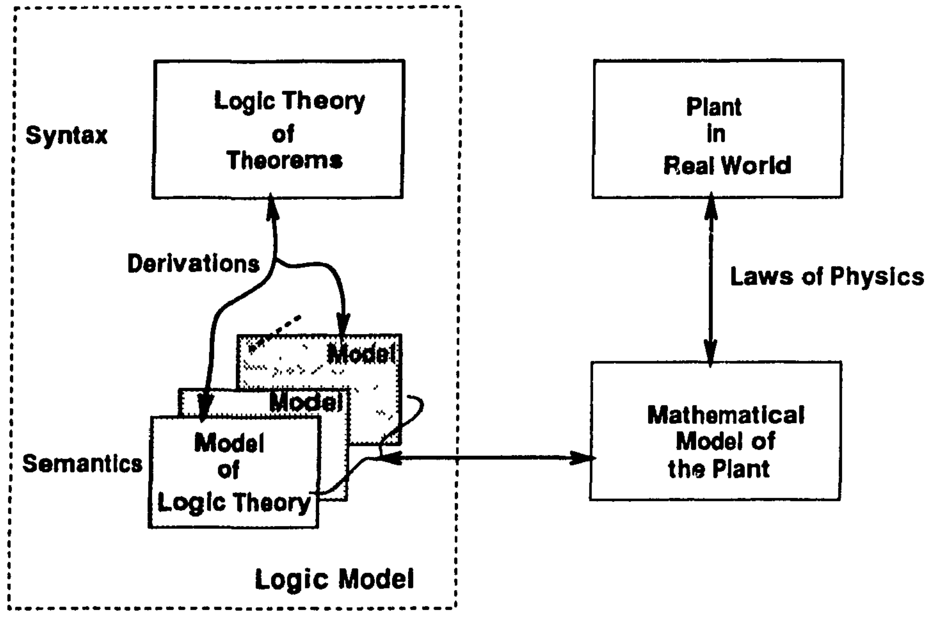


Figure 4 1. Logic-based Modeling

Logic as a formalism can be used to represent (or to model) a system. Figure 4.1 illustrate how a logic is used to model a plant, where a plant is usually modeled mathematically according to certain laws of physics. A logical theory determined by a set of axioms Σ and its associated derivation process is characterized by the set of all theorems. These theorems are constructed so that they are exactly the set of formulas which are true in every model of the logical theory. Properties of the plant are formulated in terms of the true formulas of the logic and therefore can be generated by the derivation process. Mathematical models of the plant are connected with the logic models of the logical theory so that one can formally say that truth values of formulas are characterizations of the properties of the plant.

Logic has been used, in the past, as a formal tool in computer science [Hoa85] to model program behaviors and thence to reason the correctness of the concerned program To model a dynamical system, e.g , the behavior of a concurrent program, Manna and Pnueli [MP81] proposed the use of *temporal logic* A temporal logic is a modal logic where a *necessity* and a *possibility* operators are introduced in the syntax and an *accessibility* relation among possible worlds is introduced in the semantics Ostroff and Wonham [OW85] have adopted the temporal logic to the modeling and reasoning of the correctness of the closed-loop behaviors of systems modeled by *extended state machine* (an extension of finite state machine).

4.2 Logic-based Control System

A logic-based control system here we mean an embedded logic system which provides, in real time, the control signals to the plant.

As we treated in Part I, a plant is taken to be an input-state-output finite machine An observer and a controller is a dynamical system which takes observations from the plant and outputs controlled inputs to the plant. These input and output data will flow in as *new axioms* and out as newly proved *theorems* of a logic-based control system Logics used to model and reason about plants cannot directly used to be as a logic-based control system since first axioms are fixed and hence no new axioms can be accepted in (discrete) *real* or *system* time Second, no controlled inputs can be generated as theorems since the only theorem that can be proved are hypothetical statements which are adequate for the correctness proof but not for controls

Most existing formal systems share a common feature, namely, they are all static or time-invariant with respect to the system time Even the so called temporal logic, see [Gol87] and its use in verifying the correctness of concurrent programs, see [MP81] cannot, as it stands, deal with a situation where new axioms are accepted at each system (i e real) time clock instant. Moreover, temporal logic is a modal logic wherein a statement is true at a given instant will be true at all system time clock instants and hence no *error correction* is permitted or possible.

In contrast to the notion of system time, we shall take (discrete) *logic time* to be the time that is measured between two system time clock instants and which is such that each inferential step taken in (or by) the logical deduction process consumes one unit of logic time. The following figure 4.2 shows the relationship between the logic time and the system time as we mentioned above.

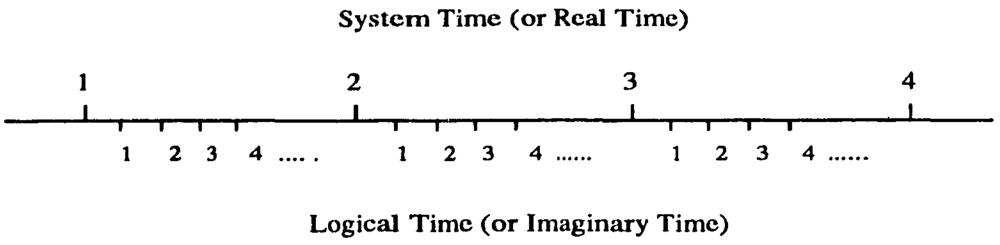


Figure 4.2 Logic Time and System Time

In the following, we give definitions of the concepts of time invariant and time varying properties of a logic system, which will be the first step towards a definition of a logic based dynamical system.

We take L to denote a set of wffs, ζ to be a set of axioms in L and we assume the set of inference rules be fixed. Intuitively, we may say a logic is (system) time invariant if the set of axioms is fixed, and hence the set of theorems is time independent. Actually, this condition is only sufficient not necessary. We may further exploit a necessary and sufficient condition for this definition by considering the equivalence classes over the set of all possible axioms as follows.

$\mathrm{Th}(\zeta) = \{\omega \,.\, \omega \in L, \zeta \vdash \omega\}$ is the set of all theorems (or the range of the deductive closure operator) derivable in L from ζ. This deductive closure operator $\mathrm{Th}{:}2^L \to 2^L$ gives rise to an equivalence relation $\equiv_{\mathrm{Th}}$ over the power set 2^L of L. This relation $\equiv_{\mathrm{Th}}$ can be defined for any $\zeta_1, \zeta_2 \in 2^L$, such that $\zeta_1 \equiv_{\mathrm{Th}} \zeta_2$ if and only if $\mathrm{Th}(\zeta_1) = \mathrm{Th}(\zeta_2)$. Furthermore, the equivalence class of ζ induced by $\equiv_{\mathrm{Th}}$ will be denoted by $[\zeta]_{\mathrm{Th}}$. In the following, we define a time invariant property for a logic system as follows

Definition 4.2.1 A logic L is said to be *time invariant,* w.r.t system time if and only if the set of axioms ζ of L is invariant, w.r.t. system time up to the equivalence class $[\zeta]_{th}$ L will be said to be a *time varying* logic if otherwise □

Most classical and contemporary logic systems have the time invariant property since they all have a fixed set of axioms and a fixed set of inference rules Examples are given by the logics used in computer science for reasoning about program behaviors (i.e the correctness of a sequential or concurrent program) such as Dynamic Logic, or Temporal Logic see [Gol87, Llo84, MP81] and for formal systems such as Petri-Net, Communicating Sequential Processes, see [Hoa85, Pet81].

In contrast to this, adaptive control problems addressed in SCT, and reasoning with uncertainty and machine learning in AI are all based on the systematic change of control laws or axioms and *rules of inference.* These systems shall be modeled by logic systems that actually vary with time or in the sense we shall discuss in this part of the thesis Within AI, these issues have been addressed formally by the construction of formal systems such as default logic see [Rei80], or non-monotonic logics in general, see [MD80, Moo85] These are intended to capture certain features of common sense reasoning or reasoning under uncertainty.

4.3 Logic Based Dynamical Systems

The general idea of a logic based dynamical system (LDS), is inspired by that of a dynamical system, that is a non-anticipative mapping ξ from the input (time) function space $\mathcal{U}$ to the output (time) function space $\mathcal{Y}$

We denote $Q(t)$ to be a set of wffs in L at the time t, they may represent a set of objects we are interested in.

Definition 4.3.1 A *logic based dynamical system (LDS)* consists of a septuple (FR(L), LA, DyA, DaA, Linf, Dinf, Q) given by a set of formation rules FR(L), axioms (LA, DyA, DaA) and rules of inference (Linf, Dinf) such that the output map is given by the query map Q. □

Namely, a LDS consists of a sequence of the following axiomatic scheme:

1. Formation Rules – FR(L)
2. Axioms
   - 2.1 Logical Axioms – LA
   - 2.2 Dynamical Axioms – DyA
   - 2.2 Data Axioms – DaA
3. Inference Rules
   - 3.1 Logical Inference Rules – Linf
   - 3.2 Dynamical Inference Rules – Dinf

A *base level query map* of $Q(t)$ to a LDS at the time t is defined to be a time dependent set $Y(t)$ such that $Y(t) = \{\omega \in L : \omega \in \text{Th}_t$ and $\omega \in Q(t)\}$, (or $Y(t) = \{\omega \in L : \omega \in \text{PTh}_t$ and $\omega \in Q(t)\}$), i.e., the intersection of Th_t (or PTh_t) with $Q(t)$. The key point here is that a base level query map is a *matching process* not a *deduction process*. Hence we have the following definition on the output function of a LDS.

Definition 4.3.2 An *output function* $Y(t)$ of a LDS L_ξ is a base level query map of $Q(t)$ to L_ξ. □

The formulation of the concept of a logic based dynamical system (LDS) led to the construction of Input/Output spaces, defined as a set of time functions on the power set of a set of wffs of a time varying logic L, and a mapping from the input space $\mathcal{U}$ to the output space $\mathcal{Y}$. So for a given time varying logic L, we have the following definition

Definition 4.3.3 An *input-output logic based dynamical system* L_ξ of the time varying logic is a triple $L_\xi = (\mathcal{U}, \mathcal{Y}, \xi)$ where

$$\mathcal{U} \underline{\triangle} \{U : \mathcal{Z}_+ \longrightarrow 2^L\} \equiv \text{ input space}$$

$$\mathcal{Y} \underline{\triangle} \{Y : \mathcal{Z}_+ \longrightarrow 2^L\} \equiv \text{ output space}$$

$$\xi : U_0^t = (u_0, u_1, \cdots u_t) \longmapsto Y_0^t = (y_0, y_1, \cdots y_t)$$

which can also be written as $\xi, U_0^t \vdash Y_0^t$. □

From the input-output system point of view, we have the schema as shown in Figure 4.3.

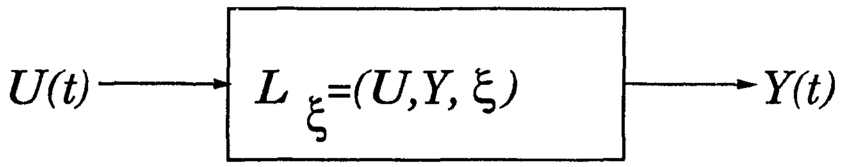


Figure 4.3: Logic Based Dynamical System

Where the input function $U(t)$ denotes a sequence of data axioms or dynamical axioms received by L_ξ up to the time t and the output function $Y(t)$ will represent a base level query of some $Q(t)$ to L_ξ.

4.4 Classical and Logic Based Observability

This subsection first introduces the concepts of a logic based dynamical observer LDO and a convergent LDO and then proves that a finite machine $\mathcal{M}$ is observable if and only if the logic-based dynamical observer $LDO(\mathcal{M})$ converges in finite time. First we need to give a mathematically precise definition of what we mean by a logic-based dynamical observer (LDO).

Definition 4.4.1 (Logic-based Dynamical Observer) Let $\mathcal{M} = (X, U, Y, \Phi, \eta)$ be a finite machine, a *logic-based dynamical observer* $LDO(\mathcal{M}) = \{FOT_O(\mathcal{M})(o_1^k) : k \in \mathcal{Z}_+\}$ *of* $\mathcal{M}$ is a tree of families of first order theories indexed by time k, where a member of a family with index k is denoted by a 4-tuple $FOT_O(\mathcal{M})(o_1^k) = (FR(\mathcal{M}), LA, DyA(\mathcal{M}), ObA(o_1^k))$ where:

$FR(\mathcal{M})$	$\underline{\Delta}$	The *formation* (*i.e.*, syntactic) *rules* of $LDO(\mathcal{M})$ specifying the well formed formulas (wffs) of $FOT(\mathcal{M})(o_1^k)$.
LA	$\underline{\Delta}$	The *logical axioms* of Predicate Calculus see [Men64]
$DyA(\mathcal{M})$	$\underline{\Delta}$	The *dynamical axioms* which describe the state transition function Φ and the output function η of $\mathcal{M}$.
$ObA(o_1^k)$	$\underline{\Delta}$	A set of *observation axioms* (specifying a sequence of input-output pairs $o_1^k = [(u_{i-1}, y_i)]_{i=1}^k \in (\phi \times Y) \times (U \times Y)^{k-1}$).

□

In the following, we take $L(\mathcal{M})$ as the language, defined by $FR(\mathcal{M})$ the formation rules for the tree of families of first order theories in a logic-based dynamical observer $LDO(\mathcal{M})$ for a given finite machine $\mathcal{M}$ We claim L should at least contain the following predicates (with or without variables, since a finite machine can be fully expressed by a propositional calculus but for the flexibility of the language we shall take predicate calculus instead of propositional calculus).

$Eq(\overline{\Phi}(x,u), x')$	: stands for $\Phi(x,u) = x'$
$Eq(\overline{\eta}(x), y)$	: stands for $\eta(x) = y$
$Eq(u_k, u^i)$	: stands for the control at the instance k is $u(k) = u_i$
$Eq(y_k, y^i)$	: stands for the observation at the instance k is $y(k) = y_i$
$eISE(x^i)$	: stands for x^i is a member of initial state estimate

$eCSE(x^i)$: stands for x^i is a member of current state estimate

Consider any member of a tree of families of first order theories, *i.e.*, $FOT_O(\mathcal{M})(o_1^k)$, a first order logical theory in a logic-based dynamical observer $LDO(\mathcal{M})$. It is uniquely determined by the given finite machine $\mathcal{M}$ and the observation sequence o_1^k in terms of the corresponding predicates. Since these observation sequences constitute a tree structure, see Chapter 2, and hence we have a tree of families of first order theories associate with a $LDO(\mathcal{M})$.

Definition 4.4.2 (Initial State Convergent LDO($\mathcal{M}$)) A logic-based dynamical observer $LDO(\mathcal{M})$ is said to be *initial state convergent* if there exists a $N \in \mathcal{Z}_+$ for all $x \in X$, for all observation sequences o_1^k, where $k \geq N$, such that $eISE(x^i) \in FOT_O(\mathcal{M})(o_1^k)$ and $\neg eISE(x^j) \in FOT_O(\mathcal{M})(o_1^k)$ with each j and $j \neq i$. Or $FOT_O(\mathcal{M})(o_1^k)$ contains every wff. □

Definition 4.4.3 (Current State Convergent LDO($\mathcal{M}$)) A logic-based dynamical observer $LDO(\mathcal{M})$ is said to be *current state convergent* if there exists an $N \in \mathcal{Z}_+$ such that for any observation sequence o_1^k, with any $k \geq N$, there exists a $x^i \in X$ such that $eCSE(x^i) \in FOT_O(\mathcal{M})(o_1^k)$ and $\neg eCSE(x^j) \in FOT_O(\mathcal{M})(o_1^k)$ with each j and $j \neq i$. Or $FOT_O(\mathcal{M})(o_1^k)$ contains every wffs. □

The above definitions probably require explanation because at first sight it appears contradictory: if the system $\mathcal{M}$ is initial state observable how can contradictory statements possibly be proved about the state? The answer is that in this definition the observation sequences o_1^k, for any $k \in \mathcal{Z}_+$, that form a part of the axiomatic system of $FOT_O(\mathcal{M})(o_1^k)$ do not necessarily give a consistent logical theory when taken together with the other axioms of $FOT_O(\mathcal{M})(o_1^k)$. If they do give a consistent logical theory then one can prove in $FOT_O(\mathcal{M})(o_1^k)$ true statements about the values of the states of $\mathcal{M}$ that could generate

o_1^k; if they do not give consistent logical theory, then in $FOT_O(\mathcal{M})(o_1^k)$ all statements (including their negations) can be established. Either way, the situation (for initial state estimates) is that for $k < N$, one cannot, in general, prove the predicate $eISE(x)$ and $\neg eISE(x')$ for all $x' \neq x$. But for $k \geq N$, all sequences o_1^k reveal themselves to be either (i) logically consistent when taken together with the observation axioms of $FOT_O(\mathcal{M})(o_1^k)$ (this intuitively corresponds to them being generated by $\mathcal{M}$ with some initial state x) or (ii) logically inconsistent. Notice that when $FOT_O(\mathcal{M})(o_1^k)$ is logically consistent we are able to prove that some sequence x_1^k of state is a consistent state sequence (with respect to o_1^k) in the sense of Definition 2 2.

Finally, in this connection, we remark that the motivation for defining an $LDO(\mathcal{M})$ that can operate on inconsistent observation strings lies in the fact that we wish to define observers (classical or logical) on their own — independent of the existence of a system generating the observations This idea is familiar in SCT (see e g. [Cai88]), where a Kalman filter may be used to process signals emitted by a system for which the filter was not designed. The curious difference between the SCT and AI situations is that the SCT filter (observer, etc.) will continue to operate in such circumstances, but will generate inaccurate (in the sense of suboptimal) estimates, on the other hand, a logic-based system may generate a contradiction and — unless otherwise designed — will jam in the sense that it can then prove all statements. These problems are the source of the studies of robustness in SCT and reasoning under uncertainty in AI.

By use of the natural interpretation of the finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ with initial state x_0 and observation sequence o_1^k, we obtain a model $(x_0, \mathcal{M}, o_1^k)$ for the first order logical theory $FOT_O(\mathcal{M})(o_1^k)$. An extremely important fact about this construction is that unless $FOT_O(\mathcal{M})(o_1^k)$ is a consistent logical theory there will not exist a set of models $\{(x_0, \mathcal{M}, o_1^k) : x_0 \in X' \subset X\}$ for the logical theory On the other hand, starting from a model $(x_0, \mathcal{M}, o_1^k)$, where o_1^k is generated by $\mathcal{M}$ with initial state x_0, the resulting logical theory $FOT_O(\mathcal{M})(o_1^k)$ will always be consistent. In the following, we shall show the equivalence between observability and the existence of a convergent LDO for a finite machine $\mathcal{M}$, by exploiting the notion of a model.

Theorem 4.4.1 ($\mathcal{M}$ is Observable iff LDO($\mathcal{M}$) is Convergent) Let $\mathcal{M} = \{X, U, Y, \Phi, \eta\}$ be a finite machine and let $LDO(\mathcal{M})$ be the logic-based dynamical observer of $\mathcal{M}$. The following statements are equivalent:

(i) $\mathcal{M}$ is initial (respectively current) state observable.

(ii) $LDO(\mathcal{M})$ is initial (respectively current) state convergent.

Proof

For simplicity we only deal with the initial case since the current case is identical.

($\Longrightarrow$)

If $\mathcal{M} = (X, U, Y, \Phi, \eta)$ is initial state observable then there exists $N = |X|^2 \in \mathcal{Z}_+$ such that for all $x' \in X$ there exists an observation sequence o_1^k with $k \geq N$, the initial state estimate $\widehat{\{x_1\}}(o_1^k) = \{x'\}$ is a singleton. Hence for any $x \in X$ the predicate $\neg eISE(x)$ is true but in particular the predicate $eISE(x')$ will be true within the model $(x', \mathcal{M}, o_1^k)$ for which the logic $FOT_O(\mathcal{M})(o_1^k)$ is based on. However by the *completeness* of the first order logical theory $FOT_O(\mathcal{M})(o_1^k)$, we can prove every true formula (but, by consistency, not every formula), and so $LDO(\mathcal{M})$ is initial state convergent.

($\Longleftarrow$)

Consider $\mathcal{M} = (X, U, Y, \Phi, \eta)$ starting from any initial state $x^* \in X$ and consider $o_1^k(x^*)$ for $k \geq N = |X|^2$. The hypothesis that $LDO(\mathcal{M})$ is initial state convergent implies that either $eISE(x^*)$ and $\neg eISE(x') \in FOT_O(\mathcal{M})(o_1^k)$ with $x^* \neq x'$, for all x', or $FOT_O(\mathcal{M})(o_1^k)$ contains every wffs.

Notice $o_1^k(x^*)$ is generated by $\mathcal{M}$. Hence a model $(x^*, \mathcal{M}, o_1^k)$ exists for $FOT_O(\mathcal{M})(o_1^k)$. This eliminates the possibility that $FOT_O(\mathcal{M})(o_1^k)$ contains every wffs. Moreover, we see that $eISE(x)$ must be true within $(x^*, \mathcal{M}, o_1^k)$ for some $x \in X$, namely for $x = x^*$.

Since the model $(x^*, \mathcal{M}, o_1^k)$ exists for $FOT_O(\mathcal{M})(o_1^k)$, inference in $FOT_O(\mathcal{M})(o_1^k)$ is *sound, i.e.*, only true statements with respect to the model can be proved in this logical theory. Hence we can prove $eISE(x)$ and $\neg eISE(x')$ for all $x' \neq x$, in $FOT_O(\mathcal{M})(o_1^k)$

Further we note that (i) if $\neg eISE(x)$ is provable for all $x \in X$, we obtain the falsehood $eISE(x^*)$ within the model $(x^*, \mathcal{M}, o_1^k)$ of the logical theory $FOT_O(\mathcal{M})(o_1^k)$ (which contradicts soundness) and (ii) if $eISE(x^0)$ is provable for some $x^0 \in X$, $eISE(x)$ cannot be proved for any $x \in X$ if $x \neq x^0$, (since $FOT_O(\mathcal{M})(o_1^k)$ is consistent by the existence of a model).

We conclude that exactly one predicate of the form $eISE(x)$, $x \in X$, is provable and, hence by soundness, true; let this hold for the unique state $x^0 \in X$. But $FOT_O(\mathcal{M})(o_1^k)$ is a first order logical theory with the model $(x^*, \mathcal{M}, o_1^k)$ and hence the only consistent state $x \in X$ satisfying $x \in \widehat{\{x_1\}}(o_1^k)$ is the singleton x^0 which satisfies $x^0 = x^*$ This shows $\mathcal{M}$ is initial state observable as required. $\square$

Similarly we can define notions of a logic-based dynamical controller (LDC), a completely observed controllable LDC or a partially observably controllable LDC and a theorem which connects the notions between logic-based system and their counterpart of classical dynamical system.

Next chapter we shall introduce a system and control theoretic logic, called, a conditional observer and controller logic (COCOLOG) for finite machines. This COCOLOG explicitly further exploit the concepts introduced in this chapter.

Chapter 5

COCOLOG for Finite Machines

As introduced in Chapter 2, a *classical dynamical observer* (hereafter called a *classical observer*) is a finite machine for which observations of the input and outputs of an observed machine are taken as inputs and which generates state estimates as outputs. In Chapter 2, we developed general results on the construction and properties of such observers for current state and initial state estimates for State-Output and Input-State-Output finite machines.

Similarly, a *classical dynamical controller (or regulator)* is a finite machine for which state estimates from a classical observer are taken as inputs, and controls generated by the classical regulator constitute the outputs. We recall that our basic Theorem 3.2 6. showed that controllers are Markovian with respect to state estimates, that is to say the state estimate contains all the information in past observations needed for all (non-anticipative) control tasks and, of course, this information is generated by an input-state-output system. Furthermore, in Chapter 3, we presented results on the construction and complexity of such controllers for completely and partially observed finite machines.

In Chapter 4, we introduced the concept of a logic-based dynamical system, which included the notion of a logic-based dynamical observer and a logic-based dynamical controller. We showed, at the meta-level, that a system is *observable* or *controllable* if and only if its corresponding logic-based system is *convergent* (logically observable) or *logically controllable*.

In this chapter, we present a theory of certain families of first order logics, called *conditional observer and controller logics* (COCOLOGs), for describing and reasoning about the state estimation and controlled evolution of a given finite machine $\mathcal{M}$ We supply a semantics for each COCOLOG in terms of interpretations of controlled transitions on the tree of state estimate sets for $\mathcal{M}$. Conditional control statements (CCSs) are formulated so that (closed loop) control actions occur when specified *past measurable* (i e , past observation dependent) conditions are fulfilled. In particular, conditional statements will include commands that steer the system state from a current partially observed state (estimate) to a target state (if such a sequence of controls can be proven to exist)

There is strong motivation to use higher level or logic based controllers in situations where time varying and adaptive control problems arise. Let us suppose the observer states of an observer tree $\widehat{\mathcal{M}}_C$ have been steered to the state estimate $\{\widehat{x_k}\}$ en route to x^T Next, suppose the dynamics of $\mathcal{M}$ change to $\mathcal{M}'$ and the current state observer tree $\widehat{\mathcal{M}}_C$ changes to $\widehat{\mathcal{M}'}_C$. For a COCOLOG system the description of this change involves a trivial rewriting-plus a consistency verification-of the axioms of the COCOLOG Then one must recompute the observer sub-tree leading to x^T and the associated (new) controlled state trajectories to x^T. However, in order for a classical controller to respond to this situation it is necessary for a set of x^T-homing feedback controllers to be *precomputed* for all possible dynamics of $\mathcal{M}$. A similar (dually related) situation occurs with the specification of *sequences of control objectives* (i e., control problems)

We regard our formulation of logic based control to be original to our work on the subject, however, it should be noted that there exists a certain commonality of viewpoint between the work described here and the literature concerned with the logical verification of program correctness [MP81], the application of these ideas to systems and control theory [Ost87, Ost89, OW89a] and logic programming [Llo84].

5.1 COCOLOG: Syntax and Semantics

A COCOLOG logic system consists of a partially ordered set, or family, of first order logics. Each of these logics corresponds to a node in the observer tree of a given finite machine. The family of these individual logic systems constitutes a logic-based dynamical system (see Chapter 4) which evolves with its environment and updates its structure as time proceeds.

To be more precise, each of the logics is equipped with the observed input and output as the data axioms of the corresponding node in the observer tree and is able to make all logical inference steps based on data axioms We present this family of first order logics in terms of axiomatic theories (For an introduction to axiomatic systems see [Men64].) In the theory presented in this chapter, we say that we let our COCOLOG system run in *real time* with the observation and control tasks, meaning that we assume all sound inferences following from a given set of axioms are available instantaneously before the next clock instant. The issue of automatic theorem proving will be addressed later in part III

In this section, we start with an introduction of the COCOLOG *language* and then we will present the *syntax* and *semantics* of the "static" part of the COCOLOG, i.e., the logic corresponding to the root node in the observation tree.

5.1.1 COCOLOG Language L

The COCOLOG language consists of a set of symbols $S(L)$ and specified formation rules (or syntax). The concerning subject of COCOLOG language is the finite machine given as $\mathcal{M} = (X^{\mathcal{M}}, U^{\mathcal{M}}, Y^{\mathcal{M}}, \Phi, \eta)^1$, where $X^{\mathcal{M}}$ is the set of *states*, $U^{\mathcal{M}}$ is the set of *controls*, $Y^{\mathcal{M}}$ is the set of *output*, Φ is a state transition function, $\Phi : X^{\mathcal{M}} \times U^{\mathcal{M}} \rightarrow X^{\mathcal{M}}$ and η

[1]We use superscrip to denote the relevant finite machine, e g $X^{\mathcal{M}}$ denote the set of states of machine $\mathcal{M}$

is a state output function, $\eta : X^{\mathcal{M}} \to Y^{\mathcal{M}}$.

We first define $S(L)$ as follows:

$$S(L) = Apr_L \bigcup Fun_L \bigcup Var_L \bigcup Cons_L \bigcup Qua_L \bigcup Lco_L \bigcup \{\perp\}.$$

The component sets of $S(L)$ are defined as follows:

Constant Symbols

$$Cons_L \;=\; \left\{x^1, \cdots, x^N\right\} \bigcup \left\{y^1, \cdots, y^p\right\} \bigcup \left\{u^1, \cdots, u^m, u^*\right\} \bigcup \left\{0, 1, \cdots, k(N)\right\}.$$

where $k(N)$ is the upper bound on time [2]

Variable Symbols

$$Var_L = \left\{x, x', x'', \cdots, \right\} \bigcup \left\{y, y', y'', \cdots, \right\} \bigcup \left\{u, u', u'', \cdots, \right\} \bigcup \left\{l, l', \cdots, \right\}.$$

Where the variables are intended to be varying in different *sorts* or domains, e g , variables $x, x', x'', \cdots$ will be interpreted to represent elements in the set of states X, variables $y, y', \cdots$ will be interpreted to represent elements in the set of state output Y, and so on.

Function Symbols

$$Fun_L = \left\{U(\cdot), Y(\cdot), \overline{\Phi}(\cdot, \cdot), \overline{\eta}(\cdot), +_L(\cdot, \cdot), -_L(\cdot, \cdot)\right\}.$$

where the sort of each function symbol is defined as the follows.

$U(a)$: a is a symbol either in $\{u^1, \cdots, u^m, u^*\}$ or in $\{u, u', \cdots, \}$

$Y(a)$: a is a symbol either in $\{y^1, \cdots, y^p\}$ or in $\{y, y', \cdots, \}$.

$\overline{\Phi}(a, b)$: a is a symbol either in $\{x^1, \cdots, x^N\}$ or in $\{x, x', \cdots, \}$; and b is a symbol either in $\{u^1, \cdots, u^m, u^*\}$ or in $\{u, u', \cdots, \}$

[2] k(N) is taken to be an arbitrary large number, for example $|X|$ or $|X|^2$ since as we can see from the results presented in Chapter 2 that an initial or current state observer dag can have at most $|X|$ or $|X|^2$ non-singleton layers before it split into singleton nodes

$\overline{\eta}(a)$: a is a symbol either in $\{x^1, \cdots, x^N\}$ or in $\{x, x', \cdots,\}$.

$+_L(a, b)$ and $-_L(a, b)$: a, b are symbols either in $\{1, 2, \cdots,\}$ or in $\{l, l', \cdots,\}$.

Terms

(i) Each constant and variable symbol is a term, i.e., $Cons_L \bigcup Var_L \subseteq Term_L$

(ii) If t is a term and f is a function symbol then $f(t)$ is a term

(iii) $Term_L$ are constructed only by steps (i) and (ii) above.

Atomic Predicate Symbols

$$Apr_L = \Big\{Eq(\cdot,\cdot), Rbl(\cdot,\cdot,\cdot)\Big\}.$$

Quantifiers

$$Qua_L = \Big\{\forall\Big\}$$

Logic Connectives

$$Lco_L = \Big\{\rightarrow\Big\}$$

5.1.2 Syntax of COCOLOG L

Any *well formed formula* of L is given by the *Backus-Naur* syntactic rule, see [Gol87]:

$$A \ ::= \ \varphi(t_1, \cdots, t_n) \mid A_1 \rightarrow A_2 \mid \bot \mid \forall v A \qquad ; \ where \ \varphi(t_1, \cdots, t_n) \in Apr_L$$
$$t_1, \cdots, t_n \in Term$$

and the set of such formulas will be denoted Fma_L.

The other logical connectives $(\neg, \vee, \wedge, \longleftrightarrow)$ and quantifier $(\exists)$ are defined as follows where (and) are used wherever the meaning of the formula can be made clearer:

$$\neg A \ ::= \ A \rightarrow \bot$$

$$A_1 \wedge A_2 \ ::= \ \neg(A_1 \rightarrow A_2)$$

$$\begin{aligned} A_1 \longleftrightarrow A_2 &::= (A_1 \rightarrow A_2) \wedge (A_2 \rightarrow A_1) \\ A_1 \vee A_2 &::= \neg A_1 \rightarrow A_2 \\ \exists v A &::= -(\forall v \neg A). \end{aligned}$$

We observe that, as introduced here, L is a *multi-sort* language, where variables may vary within different domains. This complication can easily be removed by using a *sort predicate* for each variable, and by replacing each quantified formula via the following equivalence operation:

$$\forall x A(x) \equiv \forall x(X(x) \longrightarrow A(x)),$$

$$\exists x A(x) \equiv \exists x(X(x) \longrightarrow A(x)),$$

where $X(x)$ is the sort predicate for variable x, indicating the membership relation of $x \in X$. This rewriting will allow variables to vary freely within a single domain and hence we get a *single-sort* language. In the rest of the thesis, we will not distinguish between a formula and its rewritten version for reasons of simplicity.

5.1.3 Semantics of COCOLOG L

A L-structure $\mathcal{U}_L = (\mathbf{D}, I)$ is a pair, where $\mathbf{D} = \mathbf{X} \bigcup \mathbf{Y} \bigcup \mathbf{U} \bigcup \mathbf{I}_{\mathbf{k(N)}}$ is the domain of interest and I is an interpretation function defined as follows [3]

$$\begin{aligned} I(\overline{\Phi}) &= \Phi : \mathbf{X} \times \mathbf{U} \rightarrow \mathbf{X} \\ I(\overline{\eta}) &= \eta : \mathbf{X} \rightarrow \mathbf{Y} \\ I(+_L) &= +_{k(N)} : \mathbf{I}_{\mathbf{k(N)}} \times \mathbf{I}_{\mathbf{k(N)}} \rightarrow \mathbf{I}_{\mathbf{k(N)}} \\ I(-_L) &= -_{k(N)} : \mathbf{I}_{\mathbf{k(N)}} \times \mathbf{I}_{\mathbf{k(N)}} \rightarrow \mathbf{I}_{\mathbf{k(N)}} \\ I(c) &= \mathbf{c} \in \mathbf{D} \end{aligned}$$

[3]We distinguish symbols used in COCOLOG language and in the base finite machine $\mathcal{M}$ by the convention that bold face letters denote constant and variable symbols in the base finite machine. COCOLOG function symbols will be denoted by a bar over the corresponding function symbols in the base machine.

$$\begin{aligned} I(Eq) &= \left\{(\mathbf{t},\mathbf{t}') \mid \mathbf{t},\mathbf{t}' \in \mathbf{D}, \mathbf{t} = \mathbf{t}'\right\} \subseteq \mathbf{D}^2 \\ I(Rbl) &= \left\{(\mathbf{x},\mathbf{x}',\mathbf{k}) \mid \exists \mathbf{u}_1^k \in \mathbf{U}^k, \Phi(\mathbf{x},\mathbf{u}_1^k) = \mathbf{x}'\right\} \subseteq \mathbf{X}^2 \times \mathbf{I}_{\mathbf{k(N)}} \end{aligned}$$

Where the addition, $+_{k(N)}$, and subtraction, $-_{k(N)}$, over finite integers $\{\mathbf{1},\mathbf{2},\cdots,\mathbf{k(N)} +\mathbf{1}\}$ are defined by the following expressions.[4]

$$\mathbf{a} +_{k(N)} \mathbf{b} = \begin{cases} \mathbf{a}+\mathbf{b} & \text{if } \mathbf{a}+\mathbf{b} \leq \mathbf{k(N)} \\ \mathbf{k(N)}+\mathbf{1} & \text{if } \mathbf{a}+\mathbf{b} > \mathbf{k(N)} \end{cases}$$

$$\mathbf{a} -_{k(N)} \mathbf{b} = \begin{cases} \mathbf{a}-\mathbf{b} & \text{if } \mathbf{a}-\mathbf{b} \geq \mathbf{1} \\ \mathbf{k(N)}+\mathbf{1} & \text{if } \mathbf{a}-\mathbf{b} < \mathbf{1} \end{cases}$$

These finite integer arithmetical operations are necessary to express control properties in terms of integer number of steps.

A $\mathcal{U}_L$-valuation is a function $V : Var_L \to \mathbf{D}$ satisfying

$$V(v) \in \begin{cases} \mathbf{X} & \text{if } v = x \\ \mathbf{Y} & \text{if } v = y \\ \mathbf{U} & \text{if } v = u \\ \mathbf{I}_{\mathbf{k(N)}} & \text{if } v = k \end{cases}$$

and can be extended to $V : Term_L \to \mathbf{D}$ by

$$V(t) = \begin{cases} V(t) & \text{if } t \in Var_L \\ I(t) & \text{if } t \in Cons_L \\ I(f)(V(t_1),V(t_2)) & \text{if } t = f(t_1,t_2) \text{ and } f \in Fun_L \end{cases}$$

We take $V \sim_v V'$ to mean that V and V' are identical except in the value they assign to v and

$$V(v/\mathbf{x}) = V' \text{ iff } V \sim_v V' \text{ and } V'(v) = \mathbf{x}$$

[4]Here we follow the convention that $+_L$ and $-_L$ are used in logic language to denote addition and subtraction function symbols, $+_{k(N)}$ and $-_{k(N)}$ denote the function symbols in the finite machine or in the semantic model, $+$ and $-$ denote the standard integer arithmetical operations

The satisfaction relation $\mathcal{U}_L \models A[V]$, which stands for the property that a formula A satisfies a structure $\mathcal{U}_L$ under the valuation V, is defined recursively by:

$$
\begin{aligned}
\mathcal{U}_L &\models Eq(t,t')[V] \text{ iff } V(t) = V(t'),\\
\mathcal{U}_L &\models Rbl(x,x',k)[V] \text{ iff } (V(x),V(x'),V(k)) \in I(Rbl),\\
\mathcal{U}_L &\models (A_1 \rightarrow A_2)[V] \text{ iff } \mathcal{U}_L \models A_1[V] \text{ implies } \mathcal{U}_L \models A_2[V],\\
\mathcal{U}_L &\not\models \perp[V],\\
\mathcal{U}_L &\models \forall v A[V] \text{ iff for all } \mathbf{x} \in \mathbf{D}, \text{ it is the case that } \mathcal{U}_L \models A[V(v/\mathbf{x})].
\end{aligned}
$$

A formula A is *true*, written $\mathcal{U}_L \models A$, in the structure $\mathcal{U}_L$ is defined by

$$\mathcal{U}_L \models A \quad \text{iff} \quad \text{for all } V, \text{ it is the case that } \mathcal{U}_L \models A[V],$$

conversely, A is *false*, written $\mathcal{U}_L \not\models A$, is defined by:

$$\mathcal{U}_L \not\models A \quad \text{iff} \quad \text{for all } V, \text{ it is the case that } \mathcal{U}_L \not\models A[V]$$

A formula $\mathcal{A}$ is called *valid* if it is true in all structures $\mathcal{U}_L$, i.e , $\mathcal{A}$ is valid if and only if for all $\mathcal{U}_L$, $\mathcal{U}_L \models \mathcal{A}$. A formula $\mathcal{A}$ is *satisfiable* if there exists some structure $\mathcal{U}_L$ and some $\mathcal{U}_L$ valuation V such that the satisfaction relation $\mathcal{U}_L \models \mathcal{A}[V]$ holds Obviously a formula $\mathcal{A}$ is valid if and only if $\neg\mathcal{A}$ is unsatisfiable.

5.1.4 Axiomatic Theory of Th_0

A formal logic theory of a language L consists of a set of *axioms*, that is to say a set of formulas from Fma_L, which shall be required to hold in the intended models and a set of relations on Fma_L, i.e which are called the set of *inference rules*, together with concepts of a *proof* and *theoremhood*.

A general theory of finite machines is given by simply characterizing the *functional* property and the *semi-group* property on the state transition function Φ and the output function η, as explained below.

For any $\mathbf{u} \in U$, and any $\mathbf{x}, \mathbf{x}', \mathbf{x}'' \in X$, and : ' $\mathbf{y}_1, \mathbf{y}_2 \in Y$

$$\Phi(\mathbf{x}, \mathbf{u}) = \mathbf{x}' \text{ and } \Phi(\mathbf{x}, \mathbf{u}) = \mathbf{x}'' \text{ then } \mathbf{x}' = \mathbf{x}''$$

$$\eta(\mathbf{x}) = \mathbf{y}_1 \text{ and } \eta(\mathbf{x}) = \mathbf{y}_2 \text{ then } \mathbf{y}_1 = \mathbf{y}_2$$

We write a sequence $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n$ by $\mathbf{u}_1^n$ then we can express the semi-group property as follows:

$$\Phi(\mathbf{x}, \mathbf{u}_1^n) = \Phi(\Phi(\mathbf{x}, \mathbf{u}_1^{n-1}), \mathbf{u}_n)$$

for any $\mathbf{x} \in X$ and for any sequence $\mathbf{u}_1^n \in U^n$.

A general theory will be a theory true to every finite machine or a theory which can deduce formu'as true for every finite machine. This general theory can be specialized when the transition function and the output function properties are given specifically. The sequence of specialized theories will be discussed in this and next sections. In which we discuss how a sequence of theories are formed to describe and to reason about the obsei /a*ion and control tasks We first present an axiomatic COCOLOG theory of Th_0, i.e., a logical theory able to make inferences based on the knowledge possessed by the root node in an observation tree for a given finite machine. The crucial topic of further specializations of this theory, obtained by observations on the finite machine as time proceeds, are discussed in Section 5.2.

There are two sets of axioms in Th_0, one is the *logical axioms* which are a set of valid formulas (i.e., true in all models) which together with the rules of inference generate all valid formulas; the other is a set of *special axioms* which specify the true facts concerning the subject that logic describes. Correspondingly $Th(o_1^k)$ is a logical tneory that has the input and output data as data axioms added to the logical theory Th_0.

We include logical axioms and equality axioms in Th_0 shown as follows:

LOGICAL AXIOM SCHEMATA

In the following *logical* axiom schemata, $\mathcal{A},\mathcal{B},\mathcal{C}$ are any *wffs*, i.e., $\mathcal{A},\mathcal{B},\mathcal{C} \in Fma_L$.

(i) $\mathcal{A} \to (\mathcal{B} \to \mathcal{A})$ **AXM^{log}**

(ii) $(\mathcal{A} \to (\mathcal{B} \to \mathcal{C})) \to ((\mathcal{A} \to \mathcal{B}) \to (\mathcal{A} \to \mathcal{C}))$

(iii) $(\neg\mathcal{B} \to \neg\mathcal{A}) \to ((\neg\mathcal{B} \to \mathcal{A}) \to \mathcal{B})$

(iv) $\forall v \mathcal{A}(v) \to \mathcal{A}(t)$

(v) $\forall v(\mathcal{A} \to \mathcal{B}) \to (\mathcal{A} \to \forall v \mathcal{B})$ v not free in A.

Any formula having the same form as one of these logical axiom schemata will be called a logical axiom. Hence the logical axiom schemata will give rise to an infinite number of axioms.

EQUALITY AXIOM SCHEMATA

In the following *equality* axiom schemata, $\mathcal{A}$ is any *wff*, i.e., $\mathcal{A} \in Fma_L, x, x', x'' \in Var_L$ and $f \in Fun_L$ is any function symbols in COCOLOG.

(i) $Eq(x,x)$

(ii) $Eq(x,x') \to Eq(x',x)$

(iii) $Eq(x,x'') \wedge Eq(x',x'') \to Eq(x,x')$

(iv) $Eq(x,x') \to Eq(f(x),f(x'))$ *for each function symbols f in L*

(v) $Eq(x,x') \to (P(x) \to P(x'))$ *for each of the predicate symbols P in L*

Any formula having the same form as one of these equality axiom schemata will be called an equality axiom. Hence the equality axiom schemata will give rise to an infinite number of axioms.

The special axioms for a given finite machine $\mathcal{M}$ are described as follows·

FINITE MACHINE AXIOMS

For any pair of constants $\mathbf{x^i}, \mathbf{x^j} \in X^{\mathcal{M}}, \mathbf{u^i} \in U^{\mathcal{M}}$, if $\mathbf{x^j} = \Phi(\mathbf{x^i}, \mathbf{u^i})$ is satisfied by the given finite machine $\mathcal{M}$ then we have the following *dynamic* axiom:

$$Eq(\overline{\Phi}(x^i, u^i), x^j) \qquad \mathbf{AXM^{dyn}(L)}$$

The dynamic axioms state the facts specifying the state transition function of the given finite machine $\mathcal{M}$. We note that the number of dynamic axioms is equal to $|X^{\mathcal{M}}||U^{\mathcal{M}}|$.

For any pair of constants $\mathbf{x^i} \in X^{\mathcal{M}}, \mathbf{y^i} \in Y^{\mathcal{M}}$ satisfying the relation $\eta(\mathbf{x^i}) = \mathbf{y^i}$, we have the following *output* axiom:

$$Eq(\overline{\eta}(x^i), y^i) \qquad \mathbf{AXM^{out}(L)}$$

The output axioms state the facts specifying output function of $\mathcal{M}$. The number of output axioms is equal to $|X^{\mathcal{M}}|$.

The finite machine axioms given above correspond to an infinite number of models. We get a unique model (up to name change isomorphism) when we further make the specifications of $|X| = N, |Y| = p$ and $|U| = m$ in terms of axioms. Moreover, we also need to specify addition and subtraction functions for the finite domain $\{\mathbf{1}, \mathbf{2}, \cdots, \mathbf{k(N)} + \mathbf{1}\}$ of integers. These will be addressed in the Subsection 5.4.

REACHABILITY AXIOMS

We define recursively the relation of reachability by the following reachability axioms:

$$1 \quad Rbl(x', x'', l -_L 1) \wedge Eq(\overline{\Phi}(x, u), x') \longrightarrow Rbl(x, x'', l) \qquad \mathbf{AXM^{rbl}(L)}$$

$$2 \quad Eq(\overline{\Phi}(x, u), x') \longrightarrow Rbl(x, x', 1)$$

Reachability axioms specify l step reachability relation among any pair of states, i.e., $Rbl(x, x', l)$ specifies x' is reachable from x in l steps.

Rules of Inference:

R1. MODUS PONENS

$$\frac{\mathcal{A}, \mathcal{A} \to \mathcal{B}}{\mathcal{B}} \qquad ; \ where\ \mathcal{A}, \mathcal{B} \in Fma_L$$

R2. GENERALIZATION

$$\frac{\mathcal{A}}{\forall v \mathcal{A}} \qquad ; \ where\ \ v \in Var_L$$

We write $AXM^{spe}(L)$ to denote the set of special axioms of L, i.e., $AXM^{spe}(L) = \{AXM^{dyn}(L), AXM^{out}(L), AXM^{rbl}(L)\}$. We sometimes use Σ to denote $AXM^{spe}(L)$ for simplicity.

A *proof* in L is a sequence of formulas $\mathcal{A}_1, \cdots, \mathcal{A}_k$ in Fma_L where $\mathcal{A}_i$ is either an axiom or a direct consequence of previous formulas via $R1$ or $R2$. The last formula $\mathcal{A}_k$ in the sequence is called a *theorem* and $\mathcal{A}_1, \cdots, \mathcal{A}_{k-1}$ is a proof of theorem $\mathcal{A}_k$

A formula $\mathcal{A}$ is a theorem of a first order theory with equality, written $\vdash_L \mathcal{A}$, if in a proof of $\mathcal{A}$ only logical axioms and equality axioms have been involved On the other hand, $\mathcal{A}$ is called a *consequence* (or theorem) of Σ, written $\Sigma \vdash_L \mathcal{A}$, if in a proof of $\mathcal{A}$ axioms in Σ may also have been involved.

For brevity we write Th_0 for $Th_0(L)$ which stands for the set of theorems of Σ, hence we have $Th_0 = \{\mathcal{A} : \Sigma \vdash_L \mathcal{A}\}$ and we shall use the standard notation $Th_0 \vdash A$ which is customarily read as A is a *theorem* of or *provable* (*derivable*) in the theory Th_0.

A structure $\mathcal{U}_L$ of theory Th_0 is called a *model* of the theory if and only if all the axioms of Th_0 are interpreted *true* in $\mathcal{U}_L$.

Example 5.1.1 We give a simple example to illustrate a logic based control system in terms of the COCOLOG axiomatic theory Th_0.

The finite machine $\mathcal{M} = (X^{\mathcal{M}}, U^{\mathcal{M}}, Y^{\mathcal{M}}, \Phi, \eta)$ is given in Figure 5.1

Where $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3\}, \mathbf{U} = \{\mathbf{u}^1, \mathbf{u}^2\}, \mathbf{Y} = \{\mathbf{y}^1, \mathbf{y}^2\}$, $\eta(\mathbf{x}^1) = \eta(\mathbf{x}^2) = \mathbf{y}^1, \eta(\mathbf{x}^3) = \mathbf{y}^2$ and Φ is given explicitly in the graph above.

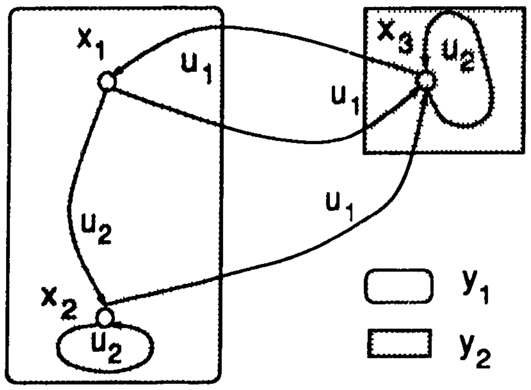


Figure 5.1: A Three State Finite Machine

The COCOLOG control system for this finite machine $\mathcal{M}$ consists of a sequence of first order theories $Th_0, Th(o_1), Th(o_1^2), \cdots$ as introduced in Chapter 4. Here we concentrate on the theory Th_0 only.

The logical axioms, the equality axioms, the axioms of reachability and the rules of inference are the same as those given earlier in this subsection, and so we will not repeat them here. In the present case, the axioms of the automaton are given explicitly as follows:

$$Eq(\overline{\Phi}(x^3, u^1), x^1)$$
$$Eq(\overline{\Phi}(x^1, u^2), x^2)$$
$$Eq(\overline{\Phi}(x^2, u^2), x^2)$$
$$Eq(\overline{\Phi}(x^1, u^1), x^3)$$
$$Eq(\overline{\Phi}(x^2, u^1), x^3)$$
$$Eq(\overline{\Phi}(X^3, u^2), x^3)$$
$$Eq(\overline{\eta}(x^1), y^1)$$
$$Eq(\overline{\eta}(x^2), y^1)$$
$$Eq(\overline{\eta}(x^3), y^2).$$

The set of theorems of Th_0 is exactly the set of true formulas of L, as is guaranteed by the general completeness result proved later in this chapter. The theorems include, as we can verify from Figure 5.1, $Rbl(x^1, x^2, 1)$, $Rbl(x^2, x^3, 1)$, $Rbl(x^1, x^3, 2)$, $\cdots$ etc

To illustrate i ɡical deduction in COCOLOG we shall give a proof of the theorem $Rbl(x^1, x^3, 2)$ in theory Th_0, which states that the state $\mathbf{x}^1$ is controllable to the state $\mathbf{x}^3$ in two steps. The proof goes as follows:

1.	$Eq(\overline{\Phi}(x^2, u^1), x^3)$	Finite Machine Axiom
2.	$\exists u, Eq(\overline{\Phi}(x^2, u), x^3)$	1 and Rule E4 see [Men64]
?.	$Rbl(x^2, x^3, 1)$	2 and $AXM^{rbl}(L)$ 1 and MP
4.	$Eq(\overline{\Phi}(x^1, u^2), x^2)$	Finite Machine Axiom
5.	$\exists u, Eq(\overline{\Phi}(x^1, u), x^2)$	4 *and Rule E4*
6.	$\exists u, Rbl(x^2, x^3, 1) \wedge Eq(\overline{\Phi}(x^1, u), x^2)$	3 *and* 5
7.	$Eq(1, 2 -_L 1)$	Arithmetic Axiom
8.	$\exists u, Rbl(x^2, x^3, 2 -_L 1) \wedge Eq(\overline{\Phi}(x^1, u), x^2)$	6,7 and Equality Axiom
9.	$Rbl(x^1, x^3, 2)$	$AXM^{rbl}(L)$, 2, *and MP*

The proofs of theorems of theory Th_0 can also be generated *mechanically* by the *resolution refutation* based theorem prover which is the subject of Part III of this thesis □

5.2 Observation Dependent COCOLOGs: $Th(o_1^k)$

A COCOLOG system is a family of first order conditional observer and controller logics, and we defined a syntax, a semantics and an axiomatic theory for Th_0. This family of logics (or logical theories) is intended to be used to reason and control the behavior of observers and controllers for a given finite machine. Corresponding to the set of paths in an observation tree, such a family of logical theories forms partially ordered set. We note that an ordered sequence of such logical theories may be viewed as being generated by a logic-based dynamical system, see Chapter 4, where a meta-level *agent* (i.e., a *logic-based dynamical system*) accepts an observation sequence $\mathbf{o}_1^k = \{(\phi, \mathbf{y}_1), (\mathbf{u}_1, \mathbf{y}_2), \cdots, (\mathbf{u}_{k-1}, \mathbf{y}_k)\}$ and generates a sequence of logical theories $Th(o_1), Th(o_1^2), \cdots, Th(o_1^k)$ with each subsequent theory $Th(o_1^k)$ being generated after the receipt of a (u_{k-1}, y_k) observation pair. Since these logical theories are observation dependent, we call them observation dependent COCOLOGs. The sequence of theories so constructed is in fact a single path in the tree of the COCOLOG family of logical theories as displayed in Figure 5.2.

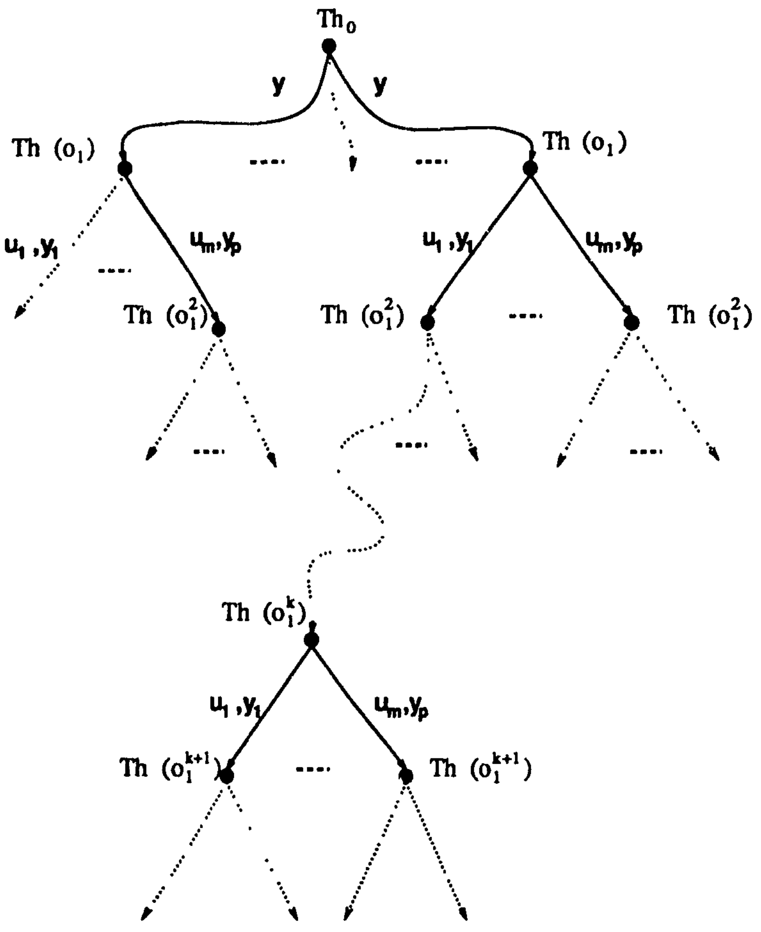


Figure 5.2: A Tree of Logics

The interaction between the finite machine and the logic-based dynamical system (or

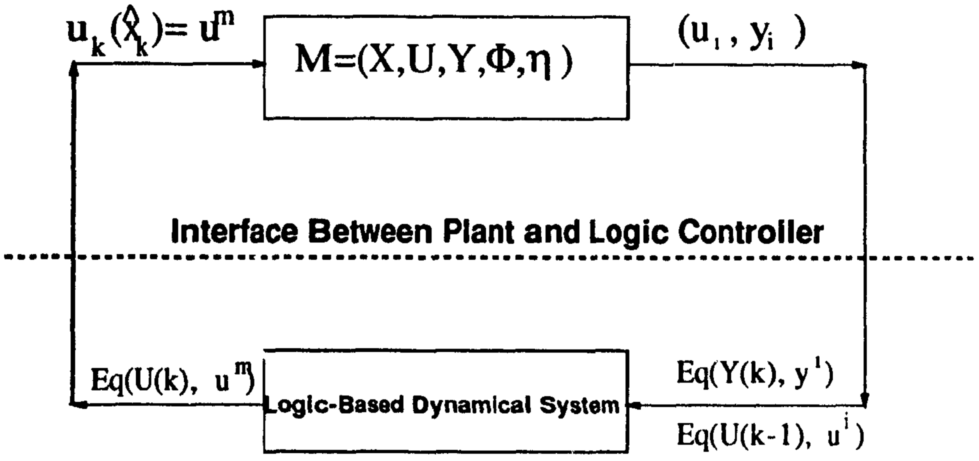


Figure 5.3: A Closed Loop Logic-Based Control System

COCOLOG to be more specific) is displayed as in Figure 5.3. Where the interface between the plant (i.e., modeled by the finite machine) and the controller (i e., the COCOLOG logics) will transform the observation sequence into new axioms and the control theorems (see below) into new control sequences.

5.2.1 COCOLOG Language $L(o_1^k)$ and Syntax

The language $L(o_1^k)$ is an extension of the language L obtained by adding new atomic predicates in the following way:

$$\begin{aligned} S(L(o_1^k)) &= S(L) \bigcup_{j=1}^{k} Apr_j, \\ \text{where} \qquad Apr_j &= \{eCSE_j(\cdot)\}. \end{aligned}$$

We define $Fma_{L_0} = Fma_L$. The set of well formed formulas $Fma_{L(o_1^k)}$ is then defined by:

$$A ::= eCSE_k(x) \mid B \mid A' \rightarrow A'' \mid \forall v A'$$

where $B \in Fma_{L(o_1^{k-1})}$, and $A', A'' \in Fma_{L(o_1^k)}$.

5.2.2 Semantics of COCOLOG $L(o_1^k)$

As before, a $L(o_1^k)$-structure, $\mathcal{U}_{L(o_1^k)} = (\mathbf{D}, I_k)$, is a pair, where the interpretation function I_k is an extension of I by:

$$I_k(eCSE_k) = \left\{\mathbf{x} : \mathbf{x} \in \widehat{\{x_k\}}(o_1^k)\right\} \subseteq \mathbf{X}.$$

The satisfaction relation of $\mathcal{U}_{L(o_1^k)} \models A[V]$ is an extension of $\mathcal{U}_{L(o_1^{k-1})} \models A[V]$ obtained by adding the following definitions:

$$\begin{aligned}\mathcal{U}_{L(o_1^k)} &\models eCSE_k(x)[V] \text{ iff } V(x) \in \widehat{\{x_k\}}(o_1^k) \text{ where } o(1) = y(1),\\ &\quad o(2) = (u(1), y(2)), \cdots, o(k) = (u(k-1), y(k))\\ \mathcal{U}_{L(o_1^k)} &\models B[V] \text{ iff } \mathcal{U}_{L(o_1^{k-1})} \models B[V] \text{ for any } B \in Fma_{L(o_1^{k-1})}\end{aligned}$$

Again the properties *true* and *false* for a formula and the concept of a *model* for a theory $Th(o_1^k)$ are defined in analogy as those in Section 5.1.3. Next we present an axiomatic theory for this logic.

5.2.3 Axiomatic Theory of $Th(o_1^k)$

We assume that, at each instant k, the observer will observe $u(k-1)$ and $y(k)$, and for each $\mathbf{u}^i$ and $\mathbf{y}^i$ such that $\mathbf{u}^i = u(k-1)$ and $\mathbf{y}^i = y(k)$, the following formulas, as observation axioms, will be used to form the axiomatic theory $Th(o_1^k)$ of $L(o_1^k)$.

OBSERVATION AXIOMS

$$Eq(Y(k), y^i) \qquad \mathbf{AXM^{out}(ob, L(o_1^k))}$$

$$Eq(U(k-1), u^i) \qquad \mathbf{AXM^{cntl}(ob, L(o_1^k))}$$

STATE ESTIMATION AXIOMS

The following is the general form of a set of *Axioms of Conditional State Estimation*, where $C_j(\cdot,\cdot)$ is a conditional formula expressible in terms of $Fma_{L(o_1^{k-1})}$ and $AXM^{out}(ob, L(o_1^k))$.

$$\begin{array}{lcll} C_1(Fma_{L(o_1^{k-1})}, Eq(Y(k), y)) & \longrightarrow & eCSE_k(x^1) & \mathbf{AXM^{est}(ob, L(o_1^k))} \\ \vdots & \longrightarrow & \vdots & \\ C_N(Fma_{L(o_1^{k-1})}, Eq(Y(k), y)) & \longrightarrow & eCSE_k(x^N). & \end{array}$$

Modus Ponens and generalization are also taken to be the rules of inference for any logical theory $Th(o_1^k)$.

Example 5.2.1 Here we take the specific state estimation strategy (which corresponds to the set theoretic state estimator given in equations 2.5 to 2.1.3 see Chapter 2). We represent it in the following axiomatic form:

In case $k > 1$:

$\mathbf{AXM^{est}(ob, L(o_1^k), +)}$

$$\exists x, eCSE_{k-1}(x) \wedge Eq(\overline{\Phi}(x, U(k-1)), x^1) \wedge Eq(\overline{\eta}(x^1), Y(k))$$
$$\longrightarrow eCSE_k(x^1)$$
$$\vdots$$
$$\exists x, eCSE_{k-1}(x) \wedge Eq(\overline{\Phi}(x, U(k-1)), x^N) \wedge Eq(\overline{\eta}(x^N), Y(k))$$
$$\longrightarrow eCSE_k(x^N).$$

$\mathbf{AXM^{est}(ob, L(o_1^k), -)}$

$$\neg\big(\exists x, eCSE_{k-1}(x) \wedge Eq(\overline{\Phi}(x, U(k-1)), x^1) \wedge Eq(\overline{\eta}(x^1), Y_k)\big)$$
$$\longrightarrow \neg eCSE_k(x^1)$$
$$\vdots$$

$$\neg\Big(\exists x, eCSE_{k-1}(x) \wedge Eq(\overline{\Phi}(x, U(k-1)), x^N) \wedge Eq(\overline{\eta}(x^N), Y(k))\Big)$$
$$\longrightarrow \neg eCSE_k(x^N).$$

In case $k = 1$;

AXM$^{\mathbf{est}}$(ob, L(o$_1$), +)

$$\begin{array}{ccc} Eq(\overline{\eta}(x^1), Y(k)) & \longrightarrow & eCSE_1(x^1) \\ \vdots & \longrightarrow & \vdots \\ Eq(\overline{\eta}(x^N), Y(k)) & \longrightarrow & eCSE_1(x^N). \end{array}$$

AXM$^{\mathbf{est}}$(ob, L(o$_1$), −)

$$\begin{array}{ccc} \neg\Big(Eq(\overline{\eta}(x^1), Y(k))\Big) & \longrightarrow & eCSE_1(x^1) \\ \vdots & \longrightarrow & \vdots \\ \neg\Big(Eq(\overline{\eta}(x^N), Y(k))\Big) & \longrightarrow & eCSE_1(x^N). \end{array}$$

□

As defined at the end of Section 5.1.3, we denote Σ as $AXM^{spe}(L)$ and here we denote Σ_k^o as $\Sigma \bigcup_{j=1}^{k}\{AXM^{out}(ob, L(o_1^k)), AXM^{cntl}\,(ob, L(o_1^k)), AXM^{est}(ob, L(o_1^k))$. We define an *Observation Theory*, $Th^o(o_1^k)$ of Σ_k^o, at the instant k by $Th^o(o_1^k) = \{\mathcal{A} : \Sigma_k^o \vdash_L \mathcal{A}\}$. Next we consider a control theory at the instant k.

CONTROL AXIOMS

The following is the general form of a set of *Conditional Control Axioms*, where $C_j(\cdot)$ is a *conditional formula* expressible in terms of $Fma_{L(o_1^k)}$.

$$C_1(Fma_{L(o_1^k)}) \longrightarrow Eq(U(k), u^1) \qquad \mathbf{AXM^{ass}(cntl, L(o_1^k))}$$

$$\begin{aligned}
\neg C_1(Fma_{L(o_1^k)}) \bigwedge C_2(Fma_{L(o_1^k)}) &\longrightarrow Eq(U(k), u^2) \\
\vdots \quad &\longrightarrow \quad \vdots \\
\bigwedge_{j=1}^{m-1} (\neg C_j(Fma_{L(o_1^j)}) \bigwedge C_m(Fma_{L(o_1^k)}) &\longrightarrow Eq(U(k), u^m) \\
\bigwedge_{j=1}^{m} (\neg C_j(Fma_{L(o_1^j)})) &\longrightarrow Eq(U(k), u^*).
\end{aligned}$$

[]

This set of axioms is central to the whole construction of COCOLOG They have the following interpretation: If the condition $C_1(Fma_{L(o_1^k)})$ is provable in the theory $Th(o_1^k)$, then invoking the first axioms, we obtain the defined constant value u^1 as the value of the control function $U(k)$; if not, but if $C_2(Fma_{L(o_1^k)})$ can be proved, then the second axiom gives the defined value u^2 to the control function $U(K)$, and so on If none of the conditions $C_1, C_2, \cdots, C_m$ hold, then the last axiom sets the control equal to the arbitrary constant u^*. This procedure uniquely determines the value of $U(K)$ When $k \to k+1$, we make the meta logical step of passing to the theory $Th(o_1^{k+1})$ carrying the constant value u^i chosen above. Then the observation axioms **AXM**$^{\mathbf{cntl}}(ob, L(o_1^{k+1}))$ state that $Eq(U(k), u^i)$. Hence, in the new $Th(o_1^{k+1})$, the observed control action $U(k)$ is precisely the constant value u^i determined in $Th(o_1^k)$.

Example 5.2.2 A set of Control Axioms which display a specific *control law*, see [CW89b] and Chapter 3, is the following:

In case of $l > 1$;

AXM$^{\mathbf{ass}}$**(cntl, L(**$\mathbf{o_1^k}$**))**

$$C_i = \forall x, \exists l, eCSE_k(x) \bigwedge Rbl(x, x', l) \bigwedge Rbl(\overline{\Phi}(x, u^i), x', l-1)$$

In case of $l = 1$;

$$C_i = \forall x, eCSE_k(x) \bigwedge Eq(\overline{\Phi}(x, u^i), x')$$

Where $1 \leq i \leq m$.

These control axioms state that if reachability from $\mathbf{x} \in \widehat{\{x_k\}}(o_1^k)$ to $\mathbf{x}^\mathbf{T}$ holds, then, by evaluating $\Phi(\widehat{\{x_k\}}(o_1^k), \mathbf{u}^\mathbf{i})$, we reduce the number of steps required to reach the predefined target state $\mathbf{x}^\mathbf{T}$ by one. In this case by an extra-logical rule we assign the state feedback control function $u(\widehat{\{x_k\}}(o_1^k), \mathbf{x}^\mathbf{T})$ to be equal to one of the calculated values of $\mathbf{u}^\mathbf{i}$. □

The *past measurable* (provable in terms of *past* data) requirement of these conditional axioms are implicitly given by the fact that the conditional statements $C_k(Fma_{L(o_1^k)})$ are expressed in terms of the language $L(o_1^k)$. The existence of the control law is forced to be true by choosing an arbitrary control u^* when other meaningful control cannot be selected.

Example 5.2.3 Continued from Example 5.2.2, we assume $x^T = x^3, l = 1$ and $Th(o_1) = Th_0 \bigcup \{Eq(y_1, y^1), AXM^{ass}(cntl, L(o_1)), AXM^{est}(ob, L(o_1), +), AXM^{est}(ob, L(o_1), -)\}$. Then there is one and only one u^1 such that $Eq(u_1, u^1)$ is a theorem of $Th(o_1)$. This fact is stated by the above existence and uniqueness axioms.

□

As we mentioned in Example 5.2.3, logical theory $Th(o_1^k)$ is defined by the axioms $\Sigma \bigcup \{AXM^{out}(ob, L(o_1^k)), AXM^{cntl}(ob, L(o_1^k)), AXM^{ass}(cntl, L(o_1^k)), AXM^{est}(ob, L(o_1^k), +), AXM^{est}(ob, L(o_1^k), -)\}$, denoted by Σ^K. The concepts of *proof* and *theorem (consequence)* in $L(o_1^k)$ are defined in the same manner as those for L. The set of all theorems of Σ^K is denoted by $Th(o_1^k)$.

5.3 Consistency and Completeness of COCOLOG

We show that the COCOLOG theory Th_0 is consistent and then prove a generalized completeness theorem stating that the set of theorems of Th_0 are exactly the set of true formulas in the class of models of $AXM^{spt}(L)$.

These results for observation dependent theories $Th(o_1^k)$ can be obtained in a same manner.

We say a set of formulas T is *consistent with respect to the axiomatic theory of first order logic with equality* if there does not exist any formula $\mathcal{A}$ where $\mathcal{A}$ and $\neg\mathcal{A}$ are both derivable from T.

The completeness of the axiomatic theory presented at Section 5 1.4 can be established by use of the following classical result on the completeness of a first order theory with equality as follows:

Theorem 5.3.1 (Henkin 1949) Every consistent set of first order formulas T has a model $\mathcal{M}_T$.

A proof of this theorem can be found in any standard book on mathematical logic, here we reference [Men64].

A first order theory with equality is any first order theory which has the equality axioms. The following theorem states a known result which states that an equality theory is complete.

Theorem 5.3.2 If a formula $\mathcal{A}$ is true in a model with equality $\mathcal{M}/=$ then it is provable in the first order logic with equality, i.e.,

$$\mathcal{M}/_= \models \mathcal{A} \text{ implies } \vdash_L \mathcal{A}.$$

A proof of this theorem can be found in any standard book on mathematical logic, here we reference [Men64].

Next consider the following generalized form of completeness theorem for the theory Th_0. As before, we denote by $\Sigma = \{AXM^{dyn}(L), AXM^{out}(L), AXM^{rbl}(L)\}$ the set of special axioms of Th_0; if $\mathcal{A}$ is a consequence of Σ under a first order theory with equality then this is written as $\Sigma \vdash_I \mathcal{A}$, and $\mathcal{M}_\Sigma$ will denote any model for Σ

Theorem 5.3.3 (Soundness) For any formula $\mathcal{A} \in Fma_L$ we have:

$$\Sigma \vdash_L \mathcal{A} \text{ implies } \mathcal{M}_\Sigma \models \mathcal{A}.$$

Proof

Soundness follows from the facts that axioms are true formulas and the rules of inference preserve truthfulness, hence all consequences or theorems of the axioms will be true in any model of the given axioms. $\Box$

A set of formulas T is *absolutely consistent* with respect to a first order theory with equality if and only if there exists some formula which is not derivable from T, i.e., $\exists \mathcal{A}, T \not\vdash_L \mathcal{A}$.

Theorem 5.3.4 (Equivalence) For any first order theory with equality where Modus Ponens is a rule of inference, the following are equivalent:

(i) T is consistent, *i.e.* $T \not\vdash_L \bot$

(ii) T is absolutely consistent, *i.e.* $\exists \mathcal{A}, T \not\vdash_L \mathcal{A}$.

Proof

(i) $\Longrightarrow$ (ii)

(i) is equivalent to the statement that for any $A, T \vdash_L A$ semantically implies $T \not\vdash_L \neg A$ (since otherwise one has $T \vdash_L A$ and $T \vdash_L \neg A$ which is $A \rightarrow \bot$ and so $T \vdash_L \bot$ by Modus Ponens).

This implies there exists a formula A such that $T \not\vdash_L A$ and hence (i) implies (ii).

(i) $\Longleftarrow$ (ii)

(i) is false if and only if for some formula A, $T \vdash_L A$ and $T \vdash_L \neg A$.

Now take any formula $B \in Fma_L$. Then we have $\vdash_L \neg A \rightarrow (\neg B \rightarrow \neg A)$ and $T \vdash_L \neg A$. Hence, by Modus Ponens, we have $T \vdash_L \neg B \rightarrow \neg A$.

In the same way, we have $\vdash_L A \rightarrow (\neg B \rightarrow A)$ and $T \vdash_L A$, hence we have $T \vdash_L \neg B \rightarrow A$. Again by Modus Ponens and the third logical axiom we get $T \vdash_L B$ and hence we get the negation of (ii) as required. □

Theorem 5.3.5 (Consistency) Σ is consistent with respect to the first order theory with equality, i.e., $\Sigma \nvdash_L \perp$.

Proof

This follows from the existence of the model $\mathcal{M}_\Sigma$ for the carefully selected set of axioms Σ. Since we know any finite machine compatible with Σ will be a model of Σ Then by the soundness theorem we will get what we want.

Take any formula $\mathcal{A} \in \Sigma$ which is an axiom of Th_0. Then we have $\mathcal{M}_\Sigma \not\models \neg\mathcal{A}$. By the soundness theorem, this implies $\Sigma \nvdash_L \neg\mathcal{A}$. By the Theorem 5.3 4, this implies $\Sigma \nvdash_L \perp$ and hence Σ is consistent. □

Now we come to the generalized completeness theorem which connects the concepts between syntax and semantics of COCOLOG theory.

Theorem 5.3.6 (Generalized Completeness) A formula $\mathcal{A} \in Fma_L$ is true in every model $\mathcal{M}_\Sigma$ of Σ if and only if $\mathcal{A}$ is a consequence of Σ under the first order theory with equality, i.e.

$$\mathcal{M}_\Sigma \models \mathcal{A} \text{ iff } \Sigma \vdash_L \mathcal{A}.$$

Proof

We only prove the *completeness* part of the theory here, the *soundness* part follows from the Theorem 5.3.3.

Suppose $\Sigma \nvdash_L \mathcal{A}$, we need to show there exists a model $\mathcal{M}_\Sigma$ such that $\mathcal{M}_\Sigma \not\models \mathcal{A}$.

$\Sigma \bigcup \{\neg\mathcal{A}\}$ is consistent since Σ is consistent and the assumption that $\Sigma \nvdash_L \mathcal{A}$. Hence by Henkin's theorem there exists a model $\mathcal{M}_{\Sigma \bigcup \{\neg\mathcal{A}\}}$ for $\Sigma \bigcup \{\neg\mathcal{A}\}$. Notice this model is also a model for Σ. Therefore we can write $\mathcal{M}_{\Sigma \bigcup \{\neg \mathcal{A}\}}$ as $\mathcal{M}_\Sigma$ with the additional condition $\mathcal{M}_\Sigma \models \neg\mathcal{A}$. This proves $\mathcal{M}_\Sigma \not\models \mathcal{A}$ as required. □

From the generalized completeness theorem we see that each theorem $\mathcal{A} \in Th_0$ will be true in every model of Σ. There are infinitely many such models. How can we determine the truthfulness of a given formula when there exists a possibly infinite number of models? By the generalized completeness theorem we know truthfulness can be identified with theoremhood. Any proof of a theorem is, by definition, a finite sequence of formulas, and in many cases the length l of the sequence is bounded by a polynomial function of the length of the theorem and axioms. However, it is known that in the worst case, a proof can be intractable (i.e., of length at least exponential in l). Specifically, Haken [Hak85] has shown recently that there are families of propositional formulas whose minimal length of resolution proofs will be bounded below by an exponential function of the number of clauses in the formulas. Moreover, the cost of searching for a proof of a true formula is greater than the length of the proof itself because no existing strategy for a mechanical theorem prover can avoid generating useless unifications or resolvents. To minimize such a cost via different strategies becomes the key issue in the subject of *mechanical theorem proving*.

Next, we construct the unique model property for Σ and therefore we get decidability of the theoremhood for COCOLOG theorems.

5.4 The Unique Model and Decidability Properties

As we mentioned before, we can get a unique model by adding additional axioms to specify sizes of $X^{\mathcal{M}}, U^{\mathcal{M}}, Y^{\mathcal{M}}$. Otherwise, there can be infinitely many different models. For example, any finite or infinite machine $\mathcal{M}' = (X^{\mathcal{M}'}, U^{\mathcal{M}'}, Y^{\mathcal{M}'}, \Phi', \eta')$ satisfying $X^{\mathcal{M}} \subseteq X^{\mathcal{M}'}, U^{\mathcal{M}} \subseteq U^{\mathcal{M}'}, Y^{\mathcal{M}} \subseteq Y^{\mathcal{M}'}$ and such that Φ' and η' are compatible with Φ and η up to $X^{\mathcal{M}}, Y^{\mathcal{M}}$ and $U^{\mathcal{M}}$, can be a model of the given machine axioms. Hence the machine axioms alone cannot uniquely characterize a given finite machine. In fact, one cannot determine a unique model by any given set of axioms. The most one can achieve by axiomatization is a set of equivalent models up to isomorphism. Hence uniqueness is only used in this sense.

Suppose $|X^{\mathcal{M}}| = N$, $|U^{\mathcal{M}}| = m$ and $|Y^{\mathcal{M}}| = p$, we first consider the *size axioms* for $X^{\mathcal{M}}$, then we can derive the size axioms for $U^{\mathcal{M}}$ and $Y^{\mathcal{M}}$ respectively in a similar manner.

The Size Axiom of $X^{\mathcal{M}}$

$$\begin{aligned}
X_N^{\mathcal{M}} :\ & \neg Eq(x^1,x^2) \wedge \neg Eq(x^1,x^3) \wedge Eq(x^1,x^4) \wedge \cdots \wedge \neg Eq(x^1,x^N) \\
& \wedge\neg Eq(x^2,x^3) \wedge \neg Eq(x^2,x^4) \wedge \cdots \wedge \neg Eq(x^2,x^N) \\
& \wedge\neg Eq(x^3,x^4) \wedge \cdots \wedge \neg Eq(x^3,x^N) \\
& \vdots \\
& \wedge\neg Eq(x^{N-1},x^N)
\end{aligned}$$

$X_N^{\mathcal{M}}$ specify the fact that there are at least N distinct constant symbols in the state space $X^{\mathcal{M}}$ of the finite machine $\mathcal{M}$, i.e., $|X^{\mathcal{M}}| \geq N$.

Next we specify the fact there are at most N elements in the intended model by $\neg X_{N+1}^{\mathcal{M}}$.

$$\neg X_{N+1}^{\mathcal{M}} : \ \forall x\Big(\bigvee_{i=1}^{N} Eq(x,x^i)\Big).$$

By adding $X_N^{\mathcal{M}}$ and $\neg X_{N+1}^{\mathcal{M}}$ to the originally proposed machine axioms the only models one can get will be the models that have exactly N distinct states. That is the set of N-state machines in which Φ and η are given as specified. Further, if we add restrictions on the size of $U^{\mathcal{M}}$ and $Y^{\mathcal{M}}$ we get a unique model for $\mathcal{M}$.

In the following we denote $\mathcal{M}$ and $\mathcal{M}'$ as finite machines and we also use them to denote the sets of elements in each machine as $\mathcal{M} = X^{\mathcal{M}} \bigcup U^{\mathcal{M}} \bigcup Y^{\mathcal{M}}$ and $\mathcal{M}' = X^{\mathcal{M}'} \bigcup U^{\mathcal{M}'} \bigcup Y^{\mathcal{M}'}$.

Definition 5.4.1 (Homomorphism) If $\mathcal{M}$ and $\mathcal{M}'$ are two finite machines, then a map h from $\mathcal{M}$ to $\mathcal{M}'$ is called a *homomorphism* if

$$\begin{aligned} h(\Phi(\mathbf{x},\mathbf{u})) &= \Phi'(h(\mathbf{x}), h(\mathbf{u})) \\ h(\eta(\mathbf{x})) &= \eta'(h(\mathbf{x})) \\ h(+_{k(N)}(\mathbf{l},\mathbf{l}')) &= +'_{k(N)}(h(\mathbf{l}), h(\mathbf{l}')) \\ h(-_{k(N)}(\mathbf{l},\mathbf{l}')) &= -'_{k(N)}(h(\mathbf{l}), h(\mathbf{l}')) \end{aligned}$$

□

Next we show the unique model property of the theory Th_0 when together with the size axioms of $X^{\mathcal{M}}$, $U^{\mathcal{M}}$ and $Y^{\mathcal{M}}$. we define

$$\begin{aligned} \Sigma^0_{\mathcal{M}} &= \Sigma \bigcup X^{\mathcal{M}}_N \bigcup \neg X^{\mathcal{M}}_{N+1} \\ &\quad \bigcup U^{\mathcal{M}}_m \bigcup \neg U^{\mathcal{M}}_{m+1} \\ &\quad \bigcup Y^{\mathcal{M}}_p \bigcup \neg Y^{\mathcal{M}}_{p+1} \end{aligned}$$

as the set of axioms for the given finite machine $\mathcal{M}$, at the instant zero.

Theorem 5.4.1 (Unique Model Property) The logical theory defined by $\Sigma^0_{\mathcal{M}}$ has a unique model up to isomorphism.

Proof

The proof of this theorem depends on the existence of a homomorphic mapping for any given pair of models of $\Sigma^0_{\mathcal{M}}$.

Now consider any two models $\mathcal{M}$ and $\mathcal{M}'$ where $\mathcal{M} = (X^{\mathcal{M}}, Y^{\mathcal{M}}, U^{\mathcal{M}}, \Phi, \eta)$ and $\mathcal{M}' = (X^{\mathcal{M}'}, Y^{\mathcal{M}'}, U^{\mathcal{M}'}, \Phi', \eta')$. By the size axioms we have $|X^{\mathcal{M}}| = |X^{\mathcal{M}'}| = N$,

$|Y^{\mathcal{M}}| = |Y^{\mathcal{M}'}| = p$ and $|U^{\mathcal{M}}| = |U^{\mathcal{M}'}| = m$. Then by the machine axioms we have $\Phi : X^{\mathcal{M}} \times U^{\mathcal{M}} \to X^{\mathcal{M}}$ and $\Phi' : X^{\mathcal{M}'} \times U^{\mathcal{M}'} \to X^{\mathcal{M}'}$; $\eta : X^{\mathcal{M}} \to Y^{\mathcal{M}}$ and $\eta' : X^{\mathcal{M}'} \times U^{\mathcal{M}'} \to X^{\mathcal{M}'}$. Now an one-to-one and onto mapping $h : \mathcal{M} \to \mathcal{M}'$ can be defined, where $\mathcal{M}$ here is also taken as the union of $X^{\mathcal{M}}$, $U^{\mathcal{M}}$ and $Y^{\mathcal{M}}$ and $\mathcal{M}'$ is also taken as the union of $X^{\mathcal{M}'}$, $U^{\mathcal{M}'}$ and $Y^{\mathcal{M}'}$.

Let L denote the set of symbols of logical theory for a finite machine, $I : L \to \mathcal{M}$ and $I' : L \to \mathcal{M}'$ be the interpretation functions correspond to the model $\mathcal{M}$ and $\mathcal{M}'$ respectively. Construct a mapping $h : \mathcal{M} \to \mathcal{M}'$ such that the following relation is satisfied:

$$h(m) = I'(I^{-1}(m)) \qquad \text{for any } m \in \mathcal{M}$$

The relations among the set of $L, \mathcal{M}, \mathcal{M}'$ and the mappings of I, I' and h are shown as follows:

$$\begin{tikzcd} & L \arrow[dl, "I"'] \arrow[dr, "I'"] & \\ \mathcal{M} \arrow[rr, "h"] & & \mathcal{M}' \end{tikzcd}$$

We need to show that h is a bijective mapping. This property is guaranteed by the bijective mappings of I and I'.

First, onto can be shown by taking any $m' \in \mathcal{M}'$, then we have $I'^{-1}(m') = l$ for some $l \in L$ and $I(l) = m$ for some $m \in \mathcal{M}$. We can show that this m is the preimage of the m' under h.

$$\begin{aligned} h(m) &= I'(I^{-1}(m)) \\ &= I'(I^{-1}(I(I'^{-1}(m')))) \\ &= I'(I'^{-1}(m')) \\ &= m' \end{aligned}$$

Second, the one-to-one property can be shown by taking any $m_1, m_2 \in \mathcal{M}$, assume that

$$h(m_1) = h(m_2) \qquad \textit{but } m_1 \neq m_2$$

The following arguments will produce a contradiction as desired:

$$\begin{aligned} h(m_1) = h(m_2) \ \text{ iff } \ & I'(I^{-1}(m_1)) = I'(I^{-1}(m_2)) \\ \text{iff } \ & m_1 = m_2 \end{aligned}$$

This shows that the mapping h as constructed is a bijective mapping.

Now if we denote $h(m) = m'$ for any $m \in \mathcal{M}$ and $m' \in \mathcal{M}'$ and take any formula $Eq(\overline{\Phi}(x^i, u^i), x^j)$ from the language L. Interpretation I will map this formula to $\Phi(I(x^i), I(u^i)) = I(x^j)$ which is $\Phi(\mathbf{x}^i_{\mathbf{m}}, \mathbf{u}^i_{\mathbf{m}}) = \mathbf{x}^j_{\mathbf{m}}$ and the interpretation I' will map the formula to $\Phi'(I'(x^i), I'(u^i)) = I'(x^j)$ which is $\Phi'(\mathbf{x}^i_{\mathbf{m}'}, \mathbf{u}^i_{\mathbf{m}'}) = \mathbf{x}^j_{\mathbf{m}'}$. Now since $h(m) = I'(I^{-1}(m))$ we have the following relationship between the two models:

$$\begin{aligned} \Phi(\mathbf{x}^i_{\mathbf{m}}, \mathbf{u}^i_{\mathbf{m}}) = \mathbf{x}^j_{\mathbf{m}} \ &\text{ iff } \ \Phi'(\mathbf{x}^i_{\mathbf{m}'}, \mathbf{u}^i_{\mathbf{m}'}) = \mathbf{x}^j_{\mathbf{m}'} \\ \eta(\mathbf{x}^i_{\mathbf{m}}) = \mathbf{y}^i_{\mathbf{m}} \ &\text{ iff } \ \eta'(\mathbf{x}^i_{\mathbf{m}'}) = \mathbf{y}^i_{\mathbf{m}'} \\ +_{k(N)}(\mathbf{l}_{\mathbf{m}}, \mathbf{l}'_{\mathbf{m}}) = \mathbf{l}''_{\mathbf{m}'} \ &\text{ iff } \ +'_{k(N)}(\mathbf{l}_{\mathbf{m}'}, \mathbf{l}'_{\mathbf{m}'}) = \mathbf{l}''_{\mathbf{m}'} \\ -_{k(N)}(\mathbf{l}_{\mathbf{m}}, \mathbf{l}'_{\mathbf{m}}) = \mathbf{l}''_{\mathbf{m}'} \ &\text{ iff } \ -'_{k(N)}(\mathbf{l}_{\mathbf{m}'}, \mathbf{l}'_{\mathbf{m}'}) = \mathbf{l}''_{\mathbf{m}'} \end{aligned}$$

From this it follows immediately that the mapping h so defined is a homomorphism, i.e., h satisfies the following relationships:

$$\begin{aligned} h(\Phi(\mathbf{x}^i, \mathbf{u}^i)) &= \Phi'(h(\mathbf{x}^i), h(\mathbf{u}^i)) \\ h(\eta(\mathbf{x}^i)) &= \eta'(h(\mathbf{x}^i)) \\ h(+_{k(N)}(\mathbf{l}, \mathbf{l}')) &= +'_{k(N)}(h(\mathbf{l}), h(\mathbf{l}')) \\ h(-_{k(N)}(\mathbf{l}, \mathbf{l}')) &= -'_{k(N)}(h(\mathbf{l}), h(\mathbf{l}')) \end{aligned}$$

Since for each dynamic axiom of the form

$$Eq(\overline{\Phi}(x^i, u^i), x^j)$$

We have $\Phi(\mathbf{x^i}, \mathbf{u^i}) = \mathbf{x^j}$ in model $\mathcal{M}$. Then the image of $\mathbf{x^j}$ under h is $h(\mathbf{x^j}) = h(\Phi(\mathbf{x^i}, \mathbf{u^j})) = \mathbf{x'^j}$. Hence we have

$$\mathbf{x^j}' = \Phi'(\mathbf{x^i}', \mathbf{u^i}') = \Phi'(h(\mathbf{x^i}), h(\mathbf{u^i})).$$

Thus

$$h(\Phi(\mathbf{x^i}, \mathbf{u^j})) = \Phi'(h(\mathbf{x^i}), h(\mathbf{u^i})).$$

Similarly we have

$$\begin{aligned} h(\eta(\mathbf{x^i})) &= \eta'(h(\mathbf{x^i})) \\ h(+_{k(N)}(\mathbf{l}, \mathbf{l}')) &= +'_{k(N)}(h(\mathbf{l}), h(\mathbf{l}')) \\ h(-_{k(N)}(\mathbf{l}, \mathbf{l}')) &= -'_{k(N)}(h(\mathbf{l}), h(\mathbf{l}')) \end{aligned}$$

Therefore $\Sigma^0_{\mathcal{M}}$ has a unique model up to isomorphism. □

Definition 5.4.2 (Proper Formula) A formula $\mathcal{P}$ is a *proper formula* with respect to a set of formula Γ if $\mathcal{P}$ contain neither any predicate symbols nor function symbols which do not appear in any formulas in Γ □

Definition 5.4.3 (Complete Axiomatization) A set of formulas Γ is said to be *complete* if either $\mathcal{P}$ or $\neg\mathcal{P}$ is a consequence of Γ for any proper formula $\mathcal{P}$ with respect to Γ □

It is known result that if a set of axioms has a unique model then that set of axioms is complete. We state this in the following theorem.

Theorem 5.4.2 The axiomatization defined by $\Sigma^0_{\mathcal{M}}$ is a complete axiomatization of $\mathcal{M}$.

We just sketch the proof as follows.

Proof

To prove that $\Sigma^0_{\mathcal{M}}$ is a complete axiomatization of $\mathcal{M}$, we need to show that for any formula $A \in L$ either $\Sigma^0_{\mathcal{M}} \vdash A$ or $\Sigma^0_{\mathcal{M}} \vdash \neg A$ is true. We know $\Sigma^0_{\mathcal{M}}$ is consitent since the existence of the models for $\Sigma^0_{\mathcal{M}}$. By Lindenbau's lemma, see [Men64] if $\Sigma^0_{\mathcal{M}}$ is consistent first order theory, then there is a consistent complete extension of $\Sigma^0_{\mathcal{M}}$. But since we know $\Sigma^0_{\mathcal{M}}$ has a unique model, see Theorem 5.4.1, this implies the complete extension of $\Sigma^0_{\mathcal{M}}$ is $\Sigma^0_{\mathcal{M}}$ and hence $\Sigma^0_{\mathcal{M}}$ is complete since otherwise $\Sigma^0_{\mathcal{M}}$ cannot have unique model. □

In the following, by *provable* we mean, first, at least, one proof of $\mathcal{P}$ exists and second, the search for any proof will always terminate.

Theorem 5.4.3 (Decidable Theoremhood) The logical theory as generated by $\Sigma^0_{\mathcal{M}}$ for any given finite machine $\mathcal{M}$ is decidable.

Proof

By the generalized completeness of first order logic in general and COCOLOG in particular, see the previous sections we know that for any formula $\mathcal{P}$, $\mathcal{P}$ is provable if $\mathcal{P}$ is a consequence of $\Sigma^0_{\mathcal{M}}$. Now for any formula $\mathcal{P}$ we start a search for all possible proofs for both $\mathcal{P}$ and $\neg\mathcal{P}$. One of these two searches will terminate since $\Sigma^0_{\mathcal{M}}$ is a complete axiomatization, i.e., either $\mathcal{P}$ or $\neg\mathcal{P}$ will be a consequence of $\Sigma^0_{\mathcal{M}}$. Thus we can conclude that the axiomatic theory generated by $\Sigma^0_{\mathcal{M}}$ is decidable.

□

Now if we denote $\Sigma^K_{\mathcal{M}} = \Sigma^0_{\mathcal{M}} \bigcup \Sigma^L$ as the axioms of the theory $Th(o^k_1)$ with the size axioms then the above results on the $\Sigma^0_{\mathcal{M}}$ can be generalized to any logical theory in a COCOLOG family.

5.5 Extra-Logical Transitions Between Logical Theories

A realization of a COCOLOG is a sequence of first order theories generated by a given sequence of observations. It corresponds to a path in the COCOLOG tree structure (see Figure 5.2). The *true* formulas in the nodes of this tree can be captured by a *possible world* interpretation of a *modal logic*, see [Gol87] Instead of modal logic, we use a family of classical first order logics to codify the state observation and control problem since we believe a modal logic representation would be too restrictive The word restrictive is used in the following two senses: First, it can easily represent a *static* world In other words, a modal logic cannot handle *unknowns* or the *changes* in the dvnamics or the environment of the system and this prohibit the use of the logic for real time control tasks. Second, it is not necessary to *code* all the *paths* of an observation tree into a logic since a physical system cannot realize all such possibilities. Therefore the extra coding of modal logic system will simply further delay its response time. For more discussion on this see [CWG88].

In order for the family of logics in a COCOLOG to work coherently, certain requirements have to be met. These requirements can be viewed as requirements on the transitions between logical theories which cannot be represented in these theories themselves. Hence the extra-logical feature of the transitions must be described at a meta-level In the following, we represent the meta-level requirement as meta-level axioms and the meta-level properties as meta-level rules of inference.

Meta-level axioms will be used to describe the assumption that there are no errors on the observation channel and the control actions sent from the logic controller will be implemented *instantly* and *correctly*. Hence there will not exist any conflict between observation and control axioms and reality.

We write $\Sigma_k^{o,c}$ as $\Sigma_k^o \bigcup_{j=1}^k \{AXM^{ass}(cntl, L(o_1^k))\}$ and define an *Observation and*

Control Theory $Th^{o,c}(o_1^k)$ of $\Sigma_k^{o,c}$ as $Th^{o,c}(o_1^k) = \{\mathcal{A} : \Sigma_k^{o,c} \vdash_L \mathcal{A}\}$. From this definition we deduce the following rule of inference to connect theories at different instants along a trajectory of observations and control actions for a finite machine.

NESTING OF THEORIES (Meta-Level Rule of Inference)

$A \in Th^o(o_1^k)$ implies $A \in Th^{o,c}(o_1^k)$, $A \in Th^{o,c}(o_1^k)$ implies $A \in Th^o(o_1^{k'})$, for any $k' > k$.

The sequence of theories satisfying the following condition

$$\cdots \subseteq Th^o(o_1^k) \subseteq Th^{c,o}(o_1^k) \subseteq Th^o(o_1^{k+1}) \subseteq \cdots$$

We see that this sequence of COCOLOGs combined with the meta-level requirements constitute a *closed loop feedback* logical control system as displayed in Figure 5.4

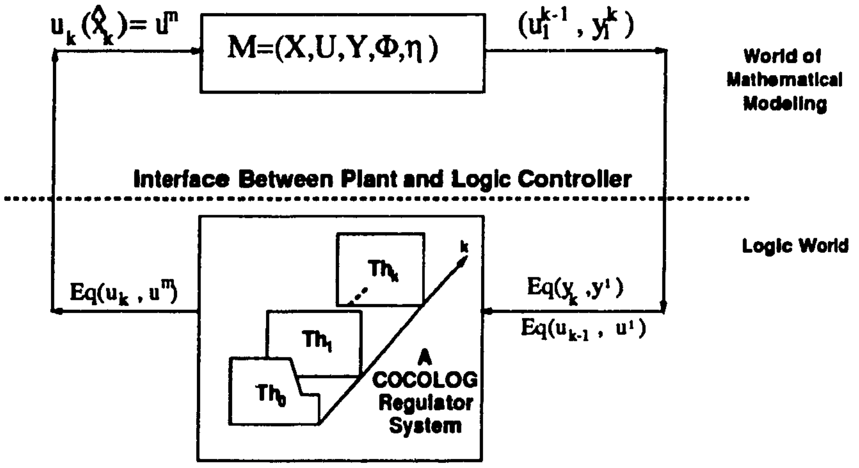


Figure 5.4: A Closed Loop Logic Control System

Example 5.5.1 Continuing with the example in Section 5.1.4, we consider here a sequence of theories $Th(o_1), Th(o_1^2), \cdots, Th(o_1^k)$. The observation dependent theories $Th(o_1^k)$ for $k \geq 1$ will contain observation dependent axioms as discussed in Section 5.2. Here we assume an observation is given as $y(1) = \mathbf{y}^1$, the target state is chosen as $x^T = \mathbf{x}^3$ and the control task is stated as: if every state $\mathbf{x}$ in the current state estimate

$\{\widehat{x_k}\}(o_1^k)$ can be steered into the target state x^T by some control in three steps then take the first control. Otherwise take the control $\mathbf{u}^*$ which will lead the machine into an idle state.

A meta-level agent will then add observation axiom $Eq(Y(1), y^1)$ and the following control axioms to theory Th_0 to generate theory $Th(o_1)$

$$C_1 \longrightarrow Eq(U(1), u^1)$$
$$\neg C_1 \bigwedge C_2 \longrightarrow Eq(U(1), u^2)$$
$$\neg(C_1 \bigvee C_2) \longrightarrow Eq(U(1), u^*)$$

Where C_i is an abbreviation of $\forall x, \exists u', u''$, $eCSE(x) \bigwedge Eq(\Phi(\Phi(\Phi(x, u'), u''), u^i), x^T)$

The theorems of the theory $Th(o_1)$ include: $eCSE_1(x^1)$, $eCSE_1(x^2)$, $\neg eCSE_1(x^3)$, $Eq(U(1), u^1), \cdots$, etc.

The control of $u(1) = \mathbf{u^1}$ will be implemented as a result of the theorem $Eq(U(1), u^1)$ proved in Th_0. The meta-level agent will then proceed to continue to build theory $Th(o_1^2), \cdots, Th(o_1^k), \cdots$. □

Part III

An Automated Reasoning Methodology for COCOLOG

In Chapter 5, we introduced the families of first order theories which we call *conditional observer and controller logic* and we gave the acronym COCOLOG to any such a family of theories. This system of theories was created for the purpose of designing observers and controllers for partially observed input-state-output finite state machines. The realizability of a logic-based control framework, as presented in Chapters 4 and 5, is based on the assumption that, at each clock instant k, the logical theory $Th(o_1^k)$ can be generated instantly, or, at least, any theorem P of the theory $Th(o_1^k)$ can be generated instantly This is indeed an idealization, Further there is the crucial implementation question of the computational complexity of ATPs (Automatic Theorem Proving)· the automatical generation of a proof for a given theorem in the propositional calculus, by resolution, can be of exponential length as a function of the length of the theorem statement (pigeonhole formula) (see [Hak85]). It is to be stressed that this statement is specific to resolution proofs. Bibel [Bib90] has shown that the connection method gives quadratic proofs of the pigeonhole formulas. The situation is even worse in the first order, semi-decidable, case, see [Men64].

In this part, we present a function evaluation based resolution, called FE–resolution, for the sets of conditional observer and controller logics. FE-resolution is, essentially, extensions of the Resolution Principle which was invented by Robinson [Rob65], and the Paramodulation technique, see [RW69, CL73] for logics with equality Completeness of the FE–resolution proof method is given, in Chapter 6, in terms of relative truthfulness and validity.

There are certain similarities between the function evaluation based resolution, presented in Chapter 6 and the procedural attachment, semantic attachment and theory resolution etc. in AI literature, see [GN87, CL73]. Recently, Myers (see [Mye91]) has presented an universal attachment technique to unify and to extend the various attachment techniques. The differences between various attachments and the FE-resolution are compared and discussed at the end of Chapter 6.

Chapter 6

Reasoning in COCOLOG with FE-Resolution

6.1 Concepts of Automated Theorem Proving

Automated theorem proving is the subject that has had the broadest and deepest impact on almost every aspects of the subject of Artificial Intelligence. Finding an effective approach to mechanically generate proofs of logical theorems has been a dream for centuries in human history. Resolution Principle invented by Robinson, see [Rob65], in 1965, is the first computer implementable theorem proving mechanism. Since then, many improvements of resolution principle have been proposed in the direction of improving the effectiveness. In this section, we present basic concepts and results related with resolution based theorem proving technique.

In Chapter 5, we present a COCOLOG system in the form of a sequence of logical theories. Each logical theory is given in the form of an axiomatic system which consists of axioms and rules of inference. In Example 5.1.1, we gave a proof of a theorem in a simple theory. It is difficult, if not impossible, to automatically generate this type of proof for a theorem of any first order theory. Since the use of *Modus Ponens* (from A and

$A \rightarrow B$, infer B) requires proving A and $A \rightarrow B$ first. There can be infinitely many clauses of the form A, $A \rightarrow B$ to be tested and this cannot be done in finite time by a machine. This situation is in analogy with the fact that in order for a human to prove a mathematical theorem it is quite often necessary to first prove some lemmas as building blocks. However, there is no effective way to teach a machine how to discover useful lemmas.

Consider a formula A of a logical theory characterized by a set of axioms Σ. In order to prove A is a theorem by a resolution based automatic theorem proving procedure, Σ is first transformed into its clausal form (this can be done effectively and automatically [CL73]), denoted by $\wedge\Sigma$, which can be viewed as a conjunction of clauses A *clause* is defined to be a disjunction of atomic predicates or negation of atomic predicates which are called *literals* (or atom), consider, for example, $Eq(x, x') \bigvee \neg Rbl(x, x', 1)$ is a clause where $Eq(x, x')$ and $\neg Rbl(x, x', 1)$ are literals. A *ground clause* is a clause where variable are replaced by constants. The same for *ground literal, ground term,* etc. By the Deduction Theorem [Men64], we have that A is a theorem of Σ if and only if $\wedge\Sigma \rightarrow A$ is a valid formula, and this holds if and only if $\neg(\wedge\Sigma \rightarrow A)$ is unsatisfiable, i e., if and only if $\wedge\Sigma \wedge \neg A$ is unsatisfiable. The unsatisfiability property has to be verified under all interpretations. Herbrand showed that there exists a special class of interpretations, called Herbrand interpretations, such that a formula is unsatisfiable if and only if it is false in all Herbrand interpretations. A *Herbrand interpretation* for a given set of clause S is considered to be the minimum interpretation possible It is constructed by building an interpretation over the set of possible ground terms (Herbrand universe) of the set of clauses S. For example, $S = \{Eq(x, x^1), Rbl(x, \overline{\Phi}(x, u), 1)\}$, then one Herbrand universe for S is $H = \{x^1, \overline{\Phi}(x^1, u^1), \overline{\Phi}(\overline{\Phi}(x^1, u^1), u^1), \cdots, \}$ Herbrand interpretation in this case could be $I = \{Eq(x^1, x^1), Rbl(x^1, x^1, 1), \cdots, \}$. A ground predicate in an Herbrand interpretation is intended to be interpreted as true in that interpretation Hence a Herbrand interpretation will contain either P or $\neg P$, where P is any ground predicate. Formal definitions of the notions of Herbrand universe and Herbrand interpretation can be found in most logic textbooks, and, in particular, for their use in mechanical theorem proving, the reader is referred to [CL73]. Herbrand interpretations makes it possible to systematically generate proofs based on the verifications of all Herbrand interpretations. The complete, possibly infinite, set of Herbrand interpretations can be organized in the form of a semantic tree. A semantic tree is a systematic structure of Herbrand interpretations. One version of the Herbrand Theorem states that a set of formulas is unsatisfiable if and only if a *finite closed semantic tree* exists, where a closed semantic tree gives rise to a set of Herbrand interpretations in which the given set of clauses will be unsatisfiable. (see [CL73]). There is a direct correspondence between a finite closed semantic tree and a resolution proof for a theorem. Completeness of resolution principle can therefore be established by this correspondence. As we pointed out before, Haken [Hak85] showed that there is a set of propositional formulas for which the length of a resolution proof can be exponential, with respect to the length of the formulas. Furthermore, in the first order case, a search for a proof of a formula can be non-terminating due to the semi-decidable property of first order logic. This corresponds to the existence of a possibly infinite number of Herbrand interpretations over which a verification can be infinite.

From the construction of a Herbrand universe and the Herbrand interpretations it may be seen that *infiniteness* of the set of interpretations is inevitable if a function symbol is introduced. Our proposed approach is to remove the logical descriptions of functions and then add the facility of function evaluation to the resolution based proof procedure. Function evaluation restricts the interpretation of the function, and hence all, unique up to isomorphism, interpretations will apply. Since by function evaluation, the model will be restricted to the constants and functions that constitute the evaluation process. We assume functions are defined deterministically and therefore only one upto isomorphic interpretation to the functions and constants is possible. Therefore the extended resolution based proof procedure will then generate only valid formulas *relative* to the functions, or in other words, will verify only the set of unsatisfiable formulas *relative* to the functions. Furthermore, this evaluability of functions actually extends to predicates and this enable us to adopt our proof procedure to *formula evaluation*.

It is commonly thought that function definition and evaluation constitute essential parts of classical, or functional programming and that in such a setting the close relationship between specification and control of data flow makes it impossible for these two parts to be separated from each other. On the other hand, the development of logic programming (PROLOG) suggests that the issue of problem solving can be viewed as a process of logical theorem proving, where the programming is decomposed into logic and control, complementary activities, see [Kow79]. The bridge connecting functional and logical programming is to be found in the way one chooses the representations of functions versus predicates. However, we shall not discuss the issue of systematically choosing such representations so as to give optimal performance in any given sense. Instead, we shall consider only the general issues of consistency and completeness of the proposed resolution proof technique and shall make some remarks concerning its complexity reduction properties.

As we pointed out earlier, the appearance of function symbols is the source of the existence of infinitely many Herbrand interpretations and hence a source of computational complexity. There are two types of function symbols in a COCOLOG, denoted $Fun = SF \cup DF$, the disjoint union of SF and DF, where SF denotes the set of *Skolem Functions* which are introduced in the process of transforming formulas into their equivalent (with respect to unsatisfiability) clausal forms and DF is the set of *Domain Functions* which are given by the special axioms of the axiomatic system under consideration.

Furthermore, we consider a partition of predicate symbols. We write $A_{pr} = EA_{pr} \cup NEA_{pr}$ to denote *evaluable atomic predicate* symbols, EA_{pr}, and *non-evaluable atomic predicate* symbols, NEA_{pr}. An evaluable predicate is a predicate for which one can effectively calculate its truth value, otherwise a predicate will be called non-evaluable. Since in a finite domain, one can always evaluate the truth value for any given atomic predicate, but one can not always do so *effectively*. Effective is used here to mean (i) a mechanical procedure exists to evaluated any given predicate; and (ii) the mechanical procedure will terminate within a polynomial time of the input predicate length. The concepts of evaluable and non-evaluable atomic predicates can be extended in a natural way to evaluable and non-evaluable *formulas*, as discussed in Section 6.3.

6.2 Resolution with Equality Theory

The Resolution Principle introduced by Robinson [Rob65] has been widely used in mechanical theorem proving. In this section we discuss mechanical theorem proof methods for COCOLOG theorems in terms of an extension, called paramodulation, of the original resolution principle for logic with equality. This is clearly of value since in a COCOLOG theory the equality predicate $Eq(\cdot,\cdot)$ is one of the atomic predicates used for expressing the machine axioms.

To prove a theorem in which the equality predicate is involved one way is to adopt the direct approach of introducing extra axioms to describe the equality relations in the logical theory. An alternative to this, which is often more efficient, can be achieved by introducing extra inference rules.

We first review how to describe an equality relation. We know that an equality is an equivalence relation and by the axioms one can substitute an equal for an equal, i.e., one can make an identity substitution. For more discussion of this see [Men64, CL73]. Truth for an equality predicate of a set of clauses S can be axiomatized in the manner of Chapter 5 and for reader's convenience we list them here denoting them $K_=(S)$:

The Equality Axiom Schemata $K_=(S)$

(1) $Eq(x,x)$

(2) $Eq(x,y) \rightarrow Eq(y,x)$

(3) $Eq(x,y) \wedge Eq(y,z) \rightarrow Eq(x,z)$

(4) $Eq(x,y) \rightarrow Eq(f(x),f(y))$ *for each function symbol f in S*

(5) $Eq(x,y) \rightarrow (P(x) \rightarrow P(y))$ *for each predicate symbol P in S*

For a resolution based theorem proving procedure, these extra axioms will certainly reduce the speed of a search or increase the size of the search space, since the number of attempted unifications during the search of a resolution proof is often an exponential function of the number of the input axioms. The number of equality axioms is a linear or at most a polynomial function of the number of function and predicate symbols To avoid such a dramatic increase in the size of the search space for resolution proofs, Robinson and Wos [RW69, Rob68] proposed a *generalized resolution principle* or *paramodulation* as an inference rule in addition to the resolution principle and we shall briefly review this idea in the following paragraphs.

To understand paramodulation, the key concepts are those of a class of equality interpretations, denoted E-interpretation and the notion of unsatisfiability in the class of equality models, denoted E-unsatisfiability. These are formally defined in terms of Herbrand interpretation.

Definition 6.2.1 (E-Interpretation) An *E-interpretation* of a set of clauses S is a Herbrand interpretation I which satisfies the following conditions.

1. $Eq(x,x) \in I$
2. $Eq(x,y) \in I$ implies $Eq(y,x) \in I$
3. $Eq(x,y), Eq(y,z) \in I$ implies $Eq(x,z) \in I$
4. $Eq(x,y) \in I, L[x] \in I$ implies $L[y] \in I$

where x, y, z are elements in the Herbrand universe of S and L is an atomic predicate or a negation of an atomic predicate, i.e. a *literal* in I. □

Definition 6.2.2 A set of clauses S is called *E-satisfiable* if there is an E-interpretation that satisfies all the clauses in S. Otherwise S is called *E-unsatisfiable.* □

An E-interpretation is an interpretation that satisfies equality axioms, i.e. that is a model of an equality theory. Next we give a theorem which states that E-interpretations indeed characterize the equality axioms.

Theorem 6.2.1 ([CL73]) Let S be a set of clauses and $K_=(S)$ be the equality axioms of S. Then S is E-unsatisfiable if and only if the set of $S \bigcup K_=(S)$ is unsatisfiable

Furthermore,

Theorem 6.2.2 ([CL73]) A finite set S of clauses is E-unsatisfiable if and only if there is a finite set S' of ground instances of clauses in S such that S' is unsatisfiable.

Proofs of the above two theorems can be found in [CL73].

Next we define paramodulation and then state the result that by using both resolution and paramodulation, we can deduce the empty clause □ from an E-unsatisfiable set of clauses. Thus the combination of paramodulation and resolution is complete for E-unsatisfiable set of clauses.

Definition 6.2.3 (Paramodulation) Let C_1 and C_2 be two clauses with no variables in common. If $C_1 = L[t] \bigcup C_1'$ and $C_2 = Eq(r,s) \bigcup C_2'$, where $L[t]$ is a literal containing the term t and C_1' and C_2' are clauses, and if t and r have a most general unifier σ, then the clause

$$L\sigma[s\sigma] \bigcup C_1'\sigma \bigcup C_2'\sigma,$$

where $L\sigma[s\sigma]$ denotes the result obtained by replacing single occurrence of $t\sigma$ in $L\sigma$ by $s\sigma$, is called a *binary paramodulant* of C_1 and C_2. We also say that we apply *paramodulation* from C_2 into C_1. □

We denote $F(S)$ as a set of reflexive functional axioms, which are any formulas in the form of either (1) or (4) of the equality axiom schemata in $K_{=}(S)$, from a given set of clauses S. In particular the clause $Eq(x,x)$ is in $F(S)$. Now we define that a deduction of resolution and paramodulation of a set of clauses S shall be applications of resolution plus paramodulation to the clauses in $S \bigcup F(S)$. Next we state the known completeness result of paramodulation.

Theorem 6.2.3 (Completeness of Paramodulation) Given a set of clauses S, S is E-unsatisfiable if and only if there is a deduction of resolution and paramodulation of the empty □ clause from S.

Proof of the completeness of paramodulation can be found in [RW69].

Consider the machine axioms in a COCOLOG theory; they are expressed in terms of equality predicate, state transition function and output function. Like the equality predicate these functions can also be expressed either by extra axioms or they can be embedded into a set of new inference rules and therefore the efficiency of the proof procedure can be improved. In our formulation the inference rules corresponding to these functions will be designed such that they can deduce the empty clause □ from any set of clauses which are unsatisfiable in any models due to the given finite machine $\mathcal{M}$ defined by state transition and output functions. We will call such models as $\mathcal{M}$-unsatisfiable More of these will be discussed in the next section.

6.3 FE-Resolution in a COCOLOG Theory

Inspired by the concepts of an E-interpretation, E-satisfiability and paramodulation for equality predicates, we propose here similar concepts for functions in a COCOLOG theory The results derived in this section can be generalized to extend to other logical theories. The key idea is to restrict the size of the Herbrand universe of a given set of clauses

Recall that a Herbrand universe of a set of clauses S is a minimum set of ground terms of S. The minimality is defined in the sense that for any interpretation I_s of S, there exists a Herbrand interpretation I_s^* of S from defined Herbrand universe such that a clause $C \in S$ is true in I_s implies that C is true in I_s^*. Therefore S is unsatisfiable if and only if S is false in all Herbrand interpretations. The point here is that a Herbrand universe is designed so that one can construct Herbrand interpretations within which the validity of a formula can be verified. In fact what we want to characterize here is a concept of relative validity with respect to the given finite machine. To be more precise, we want to define a pseudo-Herbrand universe from which we can construct interpretations and verify the validity of a formula with respect to the finite machine. Relative validity is used in the sense that a formula is valid relative to a finite machine $\mathcal{M}$ if it is interpreted as true in any model of the finite machine $\mathcal{M}$. These are semantical constructions. The corresponding syntactical counterpart will be called FE-resolution. The idea is to add a function evaluation procedure to the resolution principle and paramodulation. The syntax and the semantics will be connected by the completeness result of FE-resolution. FE-resolution can then be extended to the predicates in EA_{pr}: the set of evaluable atomic predicates at the predicate level and therefore evaluable valid formulas with respect to the models of the given finite machine can be eliminated at the formula level.

Definition 6.3.1 ($\mathcal{M}$-Universe) Given a finite machine $\mathcal{M} = (X^{\mathcal{M}}, U^{\mathcal{M}}, Y^{\mathcal{M}}, \Phi, \eta)$ and a set of clauses S, an $\mathcal{M}$-*universe* is defined as the union of the following sets:

$$\begin{aligned}
\mathcal{M}_0 &= \{a : a \text{ is a constant in } S\}, \\
\mathcal{M}_1 &= \{\Phi(a_x, a_u), \eta(a_x), +_{k(N)}(a_I, a_I'), -_{k(N)}(a_I, a_I') : \\
&\qquad a_x \in \mathcal{M}_0 \bigcap X^{\mathcal{M}}\ a_u \in \mathcal{M}_0 \bigcap U^{\mathcal{M}}, \text{ and } a_I, a_I' \in \mathcal{M}_0 \bigcap I_{k(N)}^{\mathcal{M}}\} \\
&\ \vdots \\
\mathcal{M}_{k+1} &= \{\Phi(a_x, a_u), \eta(a_x), +_{k(N)}(a_I, a_I'), -_{k(N)}(a_I, a_I') : \\
&\qquad a_x \in \mathcal{M}_k \bigcap X^{\mathcal{M}} u \in \mathcal{M}_k \bigcap U^{\mathcal{M}}, \text{ and } a_I, a_I' \in \mathcal{M}_k \bigcap I_{k(N)}^{\mathcal{M}}\}
\end{aligned}$$

Thus $\mathcal{M}$-universe is defined as

$$\mathcal{M} = \begin{cases} \bigcup_{k=0}^{\infty} \mathcal{M}_k & \text{if } \mathcal{M}_0 \neq \phi \\ X^{\mathcal{M}} \bigcup U^{\mathcal{M}} \bigcup Y^{\mathcal{M}} & \text{in case } \mathcal{M}_0 = \phi \end{cases}$$

Obviously $\mathcal{M} \subseteq X^{\mathcal{M}} \bigcup U^{\mathcal{M}} \bigcup Y^{\mathcal{M}}$ holds. □

Now we say a set A is an *atom set* (or a $\mathcal{M}$ *Base*) of S if A consists of only ground atoms occurring in S and where terms are elements of an $\mathcal{M}$-universe of S.

We define an interpretation on an $\mathcal{M}$-universe of S to be a pseudo-Herbrand interpretation where the Herbrand universe has been replaced by an $\mathcal{M}$-universe. To be more specific, let $A = \{A_1, A_2, \cdots, A_n, \cdots\}$ be the atom set of S, then

$$I = \{m_1, m_2, \cdots, m_n, \cdots\}$$

is an interpretation if m_i is either A_i or $\neg A_i$. An E-interpretation over an $\mathcal{M}$-universe will be defined as given in Definition 6.2.1 where the Herbrand Universe being replaced by the $\mathcal{M}$-universe. Next we define an $\mathcal{M}$-interpretation as follows.

Definition 6.3.2 ($\mathcal{M}$-Interpretation) Given a set of clauses S and a finite machine $\mathcal{M}$, an *$\mathcal{M}$-interpretation* is an E-interpretation which in addition satisfies the following conditions:

1. $Eq(\overline{\Phi}(a, b), a') \in I$ if $\Phi(\mathbf{a}, \mathbf{b}) = \mathbf{a}'$
2. $Eq(\overline{\eta}(a), c) \in I$ if $\eta(\mathbf{a}) = \mathbf{c}$
3. $Eq(+_L(l, l'), l'') \in I$ if $+_{k(N)}(\mathbf{l}, \mathbf{l}') = \mathbf{l}''$
4. $Eq(-_L(l, l'), l'') \in I$ if $-_{k(N)}(\mathbf{l}, \mathbf{l}') = \mathbf{l}''$

For any $\mathbf{a}, \mathbf{a}' \in X^{\mathcal{M}}$, $\mathbf{b} \in U^{\mathcal{M}}$, $\mathbf{c} \in Y^{\mathcal{M}}$ and $\mathbf{l}, \mathbf{l}', \mathbf{l}'' \in I_{k(N}^{\mathcal{M}}$ □

In fact an $\mathcal{M}$-interpretation is a pseudo-Herbrand interpretation where all machine axioms will be interpreted true and $Eq(\cdot, \cdot)$ is interpreted as the identity relation. A set of clauses S is said to be satisfiable in an interpretation I, denoted $I \models_s S$, if and only if for any clause C in S there exists some ground clause C' of C such that C' is interpreted true in I. Truthfulness of first order formulas can be defined by the standard Tarski semantics (see [Men64] for reference). Next we define $\mathcal{M}$-validity and $\mathcal{M}$-unsatisfiability.

Definition 6.3.3 ($\mathcal{M}$-Relative Validness) A formula $\mathcal{A}$ is said to be *$\mathcal{M}$-relative valid, or $\mathcal{M}$-valid* if $\mathcal{A}$ is interpreted true in every $\mathcal{M}$-interpretation, i.e. for each $\mathcal{M}$-interpretation $I_{\mathcal{M}}$ we have $I_{\mathcal{M}} \models \mathcal{A}$. □

Correspondingly we have:

Definition 6.3.4 ($\mathcal{M}$-Relative Unsatisfiability) A formula A is said to be *$\mathcal{M}$-relative unsatisfiable, or $\mathcal{M}$-unsatisfiable* if A is interpreted false in every $\mathcal{M}$-interpretation, i.e. for each $\mathcal{M}$-interpretation $I_{\mathcal{M}}$ we have $I_{\mathcal{M}} \not\models A$. □

Obviously we have A is $\mathcal{M}$-relatively valid if and only if $\neg A$ is $\mathcal{M}$-relatively unsatisfiable. Next we show that an $\mathcal{M}$-interpretation characterizes the class of models for the given finite machine $\mathcal{M}$ defined by the set of axioms $\Sigma^0_{\mathcal{M}}$. But before we do so, we first give two lemmas. It is a known result that if a set of clauses S is satisfiable in an interpretation I then S is also satisfiable in any Herbrand interpretation I^* corresponding to I (see Lemma 4.1 in [CL73]). Now we extend this result from Herbrand interpretations to E-interpretations and $\mathcal{M}$-interpretations respectively.

Lemma 6.3.1 Let $K_=(S)$ be the equality axioms for a set of clauses S, then if a Herbrand interpretation I^* satisfies $K_=(S)$ then I^* is also an E-interpretation.

Proof. By the converse of Theorem 6.2.2, $I^* \models K_=(S)$ implies that for any axiom $A \in K_=(S)$ each ground instance A' of A such that $I^* \models A'$. But since each axiom in $K_=(S)$ is assumed to be quantified universally it follows that any ground instance A' of any axiom A in $K_=(S)$ is satisfied by I^*. Therefore I^* meets the conditions for an E-interpretation and so I^* is also an E-interpretation. □

Lemma 6.3.2 Let $\Sigma^0_{\mathcal{M}}(S)$ be the machine axioms for a set of clauses S, then .or any E-interpretation I_E satisfying $\Sigma^0_{\mathcal{M}}(S)$ it is the case that I_E is also an $\mathcal{M}$-interpretation

Proof. The proof is similar to that of the previous Lemma. $I_E \models \Sigma^0_{\mathcal{M}}(S)$ implies that for any axiom $A \in \Sigma^0_{\mathcal{M}}(S)$ there exists a ground instance A' of A such that $I_E \models A'$. Since each axiom A in $\Sigma^0_{\mathcal{M}}(S)$ is universally quantified it follows the $I_E \models A$. This implies $I_E \models A'$ for any ground instance A' of A in $\Sigma^0_{\mathcal{M}}(S)$ and hence all the conditions for I_E as an $\mathcal{M}$-interpretation are met. We conclude that I_E is an $\mathcal{M}$-interpretation □

Theorem 6.3.1 Let S be a set of clauses, and let $\Sigma^0_{\mathcal{M}}(S)$ and $K_=(S)$ be the set of machine axioms and equality axioms respectively. Then S is $\mathcal{M}$-unsatisfiable if and only if $S \bigcup \Sigma^0_{\mathcal{M}} \bigcup K_=(S)$ is unsatisfiable.

Proof. Suppose S is $\mathcal{M}$-unsatisfiable but $S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$ is satisfiable. The latter implies that

$$I \models S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$$

This is equivalent to

$$I \models S \text{ and } I \models \Sigma^0_{\mathcal{M}}(S) \text{ and } I \models K_=(s)$$

This implies for any Herbrand interpretation I^* corresponding to I we have

$$I^* \models S \text{ and } I^* \models \Sigma^0_{\mathcal{M}}(S) \text{ and } I^* \models K_=(S)$$

By Lemma 6.3.1 we have

$$I_E \models S \text{ and } I_E \models \Sigma^0_{\mathcal{M}}(S)$$

By Lemma 6.3.2 we known that S is satisfied by an $\mathcal{M}$-interpretation, that is to say $I_{\mathcal{M}} \models S$, which contradicts the fact that S is $\mathcal{M}$-unsatisfiable and hence the result is proved.

The other direction of the proof is simple. Suppose $S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$ is unsatisfiable but S is $\mathcal{M}$-satisfiable. The latter statement implies there is an $\mathcal{M}$-interpretation $I_{\mathcal{M}} \models S$. Clearly we have $I_{\mathcal{M}} \models \Sigma^0_{\mathcal{M}}(S)$ and $I_{\mathcal{M}} \models K_=(S)$ since $I_{\mathcal{M}}$ is also an E-interpretation. Therefore we have $I_{\mathcal{M}} \models S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$ which contradicts to the assumption that $S \bigcup \Sigma^0_{\mathcal{M}}(S) \bigcup K_=(S)$ is unsatisfiable. This completes the proof. □

6.3.1 FE-Resolution and Its Completeness

In the following we denote $f(x_1, \cdots, x_n)$ as a n-ary function with n variables $x_1, x_2, \cdots, x_n$, and $b \equiv f(a_1, \cdots, a_n)$ shall denote function f evaluated at constant, $a_1, \cdots, a_n$ is b. Now we define rules for function evaluation based resolution, also called FE-resolution. We will show that these rules combined with rules of standard resolution and paramodulation are complete. In particular, an empty clause will be generated from a set of $\mathcal{M}$-unsatisfiable set of clause by applying these extended resolution. Function evaluation can be done either at constant level or at variable level, i.e., f can be partially evaluated at a_1 etc to give the function $f(a_1, \cdots, a_{m-1}, x_m, \cdots, x_n)$. Thus FE-resolution should also be define for constant and variable cases respectively.

Definition 6.3.5 (Constant FE-resolution) Let C_1 be a clause which is in the form of $C_1 = L[f(a_1, \cdots, a_n)] \vee C_1'$, where $L[f]$ is a literal containing the function symbol f in its ground form $f(a_1, \cdots, a_n)$, C_1' is a subclause of C_1, then infer the inferential step

$$\frac{C_1 = L[f(a_1, \cdots, a_n)] \vee C_1'}{C = L[b] \vee C_1'}$$

where $b \equiv f(a_1, \cdots, a_n)$, is called *constant FE-resolution* and the inferred clause C is called a *constant FE-resolvent of* C_1 □

Applying constant FE-resolution will result in the generation of all equivalent clause to those which contain ground terms but not ground function terms. Constant FE-resolution can be extended to variable FE-resolution as follows:

Definition 6.3.6 (Variable FE-Resolution) Let C_1 be a clause which is in the form

$$C_1 = L[f(a_1, \cdots, a_k, x_{k+1}, \cdots, x_n, y_1, \cdots, y_l)] \vee C_1'(x_{k+1}, \cdots, x_n, x_{n+1}, \cdots, x_{n+m})$$

where $L[f]$ is a literal containing the function symbol f which is in a semi-ground form, $f(a_1, \cdots, a_k, x_{k+1}, \cdots, x_n, y_1, \cdots, y_l), C_1'$ is a subclause of C_1 and C_1' has shared variables $x_{k+1}, \cdots, x_n$ with $L[f]$. If $b \equiv f(a_1, a_2, \cdots, a_n, c_1, \cdots, c_l)$ for any constants $a_{k+1}, a_{k+2}, \cdots, a_n, c_1, \cdots, c_l$, then the inferential step

$$\frac{L[f(a_1, \cdots, a_k, x_{k+1}, \cdots, x_n, y_1, \cdots, y_l)] \vee C_1'}{C = L\sigma[b] \vee C_1'\sigma}$$

where $\sigma = \{x_{k+1}/a_{k+1}, x_{k+2}/a_{k+2}, \cdots, x_n/a_n, y_1/c_1, \cdots, y_l/c_l\}$ is a substitution, is called *variable FE-resolution.* The inferred clause C is called a *variable FE-resolvent of* C_1 *Therefore the resolvent is given in the following form:*

$$C = L\sigma[b] \vee C_1'(a_{k+1}, \cdots, a_n, x_{n+1}, \cdots, x_{n+m})$$

□

A variable FE-resolution will generate a ground instance of the function and a semi-ground instance of the original clause which are logically implied by the original clause. We define a *FE-resolution step* is an inference step of either variable or constant FE-resolution by which a FE-resolvent is produced. Next we prove that both constant FE-resolution and variable FE-resolution are complete, i e , an empty clause □ will be derived by using the standard resolution, paramodulation and FE-resolution to a given set of clauses S, if S is $\mathcal{M}$-unsatisfiable. This is stated in the next theorem.

In the following, we denote σ_{re}, σ_{rp} and σ_{fe} as deduction rules of resolution, resolution and paramodulation, resolution and paramodulation and function evaluation respectively. When the empty clause □ is deducible from a base clause set S by a sequence of deductions using rules from the set σ we write by $S \vdash_\sigma □$. A deduction of the empty clause □ from a base clause set S by a σ deduction rule shall be denoted by $D[S \vdash_{\sigma} \Box]$. Again, '$\equiv$' and '$=$' are used as symbols of meta-level language to denote *function evaluation* and *represent* respectively. That is $d \equiv f(a,b,c)$ denotes: f evaluated at a, b, c is d and $C = Eq(f(x,y),z)$ denotes: C represents clause $Eq(f(x,y),z)$.

Theorem 6.3.2 (Completeness of FE-Resolution) Given a set of clauses S, S is $\mathcal{M}$-unsatisfiable, for a finite machine $\mathcal{M}$, if and only if there is a deduction of the empty clause $\Box$ from S using rules of resolution, paramodulation and FE-resolution.

Proof

By Theorem 6.3.1 we know that S is $\mathcal{M}$-unsatisfiable if and only if $S \bigcup K_{=}(S) \bigcup \Sigma^{0}_{\mathcal{M}}(S)$ is unsatisfiable. By Theorem 6.2.1 $S \bigcup K_{=}(S) \bigcup \Sigma^{0}_{\mathcal{M}}(S)$ is unsatisfiable if and only if $S \bigcup \Sigma^{0}_{\mathcal{M}}(S)$ is E-unsatisfiable. Furthermore, by the completeness of the paramodulation and Theorem 6.2.3 we have $S \bigcup \Sigma^{0}_{\mathcal{M}}(S) \vdash_{\sigma_{rp}} \Box$ and therefore there exits a deduction of resolution and paramodulation of the empty $\Box$ clause from $S \bigcup \Sigma^{0}_{\mathcal{M}}(S)$, denoted by $D[S \bigcup \Sigma^{0}_{\mathcal{M}}(S) \vdash_{\sigma_{rp}} \Box]$.

Now we need to show $S \vdash_{\sigma_{fe}} \Box$, that is, $\Box$ is deducible from S by using the deduction rules from σ_{fe}. It is enough to show the existence of a deduction $D[S \vdash_{\sigma_{fe}} \Box]$. It will be seen that this deduction can be obtained directly from the existing deduction, $D[S \bigcup \Sigma^{0}_{\mathcal{M}}(S) \vdash_{\sigma_{rp}} \Box]$ and that the changes of the base clause set from $S \bigcup \Sigma^{0}_{\mathcal{M}}(S)$ to S will be compensated for by the additional syntactic inference rules of function evaluations.

According to our earlier definition, $\Sigma^{0}_{\mathcal{M}}(S)$ consists of two set of clauses which specify: (1) functions, i.e., in the form of $\Phi(\mathbf{x_i}, \mathbf{u_i}) = \mathbf{x_j}, \eta(\mathbf{x_i}) = \mathbf{y_i}, +_{k(N)}(\mathbf{l}, \mathbf{l'}) = \mathbf{l''}$ and $-_{k(N)}(\mathbf{l}, \mathbf{l'}) = \mathbf{l''}$ and (2) variable size clauses which specify the facts $x_1 \neq x_2, x_1 \neq x_3, \cdots$. We need only consider to convert the first set of clauses to FE-resolution since the set (2) is implied by the use of function evaluation and function definition.

Consider a clause $Eq(f(a,b),c)$ in $\Sigma^{0}_{\mathcal{M}}(S)$ (the same is true for unary-function $Eq(f(a), b)$, a state output function in $\Sigma^{0}_{\mathcal{M}}(S)$). If this clause, denoted C_1 is one of the parent clauses in the given deduction process, the other parent clause must be the form of $C_2 = \neg Eq(f(x,u),y) \wedge C_2'(x,u,y)$, where $C_2'(x,u,y)$ is a subclause of C_2. In this deduction the letters x, u, y will either take the variable -type values- which we denote x, u, y or they take constant values which in this proof are denoted a, b, c, i.e., we are dealing with variable or semi-variable FE-resolution as defined in Definitions 6 3.6 and 6 3.5 Taking the element **x** to have the generic variable letter form x or the generic constant letter form **a**, we get the first line of the following table where eight cases need to be considered

Var	*Cases*							
x	a	a	x	a	x	x	a	x
u	b	b	b	x	b	y	x	y
y	c	y	c	c	y	c	y	z

In the following, we show this case by case:

Case 1. If $C_2 = \neg Eq(f(a,b),c) \bigvee C_2'(a,b,c)$ then the resolvent produced by the use of the resolution principle to this and the clause $Eq(f(a,b),c)$ in $\Sigma^o_M(S)$ is $C = C_2'(a,b,c)$.

Consider the application of function evaluation to $f(a,b)$ in C_2, we shall have $\neg Eq(c,c) \bigvee C_2'(a,b,c)$ if $c \equiv f(a,b)$. This clause can then be further resolved with $Eq(x,x)$ to the final resolvent $C' = C_2'(a,b,c) = C$. The case $c \neq c' \equiv f(a,b)$ will not occur since function evaluation will be consistent with the axiom of $Eq(f(a,b),c)$.

Case 2. If $C_2 = \neg Eq(f(a,b),y) \bigvee C_2'(a,b,y)$, then the resolvent is $C = C_2'\sigma(a,b,y\sigma)$, where σ is a substitution defined as $\sigma = \{c/y\}$.

Consider the application of function evaluation to $f(a,b)$ in C_2, again we get $\neg Eq(c,y) \bigvee C_2'(a,b,y)$ if $c \equiv f(a,b)$. Furthermore, we may resolve this with $Eq(x,x)$, and we have the final resolvent $C' = C_2\sigma(a,b,y\sigma)$ where $\sigma = \{c/x, c/y\}$. Obviously $C' = C$.

Case 3. If $C_2 = \neg Eq(f(x,b),c) \bigvee C_2'(x,b,c)$ then the resolvent is $C = C_2'\sigma(x\sigma,b,c)$, with $\sigma = \{a/x\}$.

Consider the application of function evaluation to $c \equiv f(x,b)$. We shall find some a' such that $c \equiv f(a',b)$ and define $\sigma = \{a'/x\}$. Now this function evaluation will lead us to $\neg Eq(c,c) \bigvee C_2'\sigma(x\sigma,b,c)$. Again resolve this with $Eq(x,x)$ we get $C' = C_2'\sigma(x\sigma,b,c) = C$.

Case 4. $C_2 = \neg Eq(f(a,x),c) \bigvee C_2'(a,x,c)$ is treated in a similar manner as in Case 3.

Case 5. If $C_2 = \neg Eq(f(x,b),y) \bigvee C_2'(x,b,y)$ then the resolvent is $C = C_2'\sigma(x\sigma,b,y\sigma)$, with $\sigma = \{a/x, c/y\}$.

Consider the application of function evaluation to $y \equiv f(x,b)$ we shall find some a', c' such that $c' \equiv f(a',b)$ and define $\sigma = \{a'/x, c'/y\}$. Now this function evaluation will result $\neg Eq(c',c') \bigvee C_2'\sigma(x\sigma,b,y\sigma)$. Further resolve this with $Eq(x,x)$ and get $C' = C_2'\sigma(x\sigma,b,y\sigma) = C$.

Case 6. $C_2 = \neg Eq(f(x,y),c) \bigvee C_2'(x,y,c)$ is treated in a similar manner as in Case 5.

Case 7. $C_2 = \neg Eq(f(a,x),y) \bigvee C_2'(a,x,y)$ is treated in a similar manner as in Case 5.

Case 8. If $C_2 = \neg Eq(f(x,y),z) \bigvee C_2'(x,y,z)$ then the resolvent is $C = C_2'\sigma(x\sigma,y\sigma,z\sigma)$ with $\sigma = \{a/x, b/y, c/z\}$.

An application of function evaluation to $z \equiv f(x,y)$ may lead us to any a', b', c' such that $c' \equiv f(a',b')$, but not necessarily with $a' = a$ and $b' = b$ and $c' = c$.

This completes our case by case analysis and hence we can conclude that the existence of a deduction of the form $D[S \bigcup \Sigma^0_{\mathcal{M}}(S) \vdash_{\sigma_{rp}} \Box]$ implies the existence of a deduction of the form $D[S \vdash_{\sigma_{fe}} \Box]$ and hence we conclude that FE-resolution is complete.

$\Box$

In cases, except in Cases 1 and 2, trial and error steps are inevitable in order to get a substitution identical to the resolvent produced by the resolution and paramodulation.

6.3.2 Literal and Formula Evaluation

This subsection extends the FE-resolution introduced in previous subsection to literal and subclause evaluation respectively.

To evaluate literals or subclauses, one shall expect to receive logical truth values as the result of evaluation. These truth values can be varying from *true, false* or *unknown* depending on the nature of the literal or the subclause to be evaluated. For example, $Eq(x,x)$ will be evaluated to be logical true and $\neg Eq(x,x)$ will be evaluated to be logically false in any models in which $Eq(\cdot,\cdot)$ is interpreted as equality relation. While some predicate may result in an unknown truth value especially when infinite model is the case. We know, in particular, that COCOLOG has unique model and is decidable Therefore, any formula in a COCOLOG languag‹ ,will have a deterministic truth value and the same for the evaluation process. Here, we only consider the evaluation being applied to the ground literals which could be resulteα from some FE-resolution step, or a FE-resolvent.

Definition 6.3.7 (Literal Evaluation) Let C_1 be a clause which is in the form of

$$C_1 = L[f(x)] \vee C_1'(x)$$

or

$$C_1 = L \vee C_1'(x),$$

where L does not contain any other shared variables with C_1' except in the first case that variable x is shared through the function symbol f. Then if $L[b]$ or L is evaluated to be false, where $b \equiv f(a)$ for some a, the inferential step

$$\frac{L[f(x)] \vee C_1'(x)}{C = C_1'(a)} \quad \text{or} \quad \frac{L \vee C_1'(x)}{C = C_1'(x)}$$

is called the *literal evaluation or LE-resolution* C is called *LE-resolvent.* □

This is obviously a sound inference step, since false in disjunction with any formula does not change the truth value of that formula, i.e., $\perp \vee C$ is logically equivalent to C. The completeness of this step can be seen through the following argument. If $\perp \vee C$ can produce an empty clause $\square$ (or $\perp$) through $\perp \vee C$ with $\neg C$ then C with $\neg C$ can also produce the empty clause $\square$, according to resolution principle.

The COCOLOG system equipped with FE-resolution constituts a function -logic-function closed loop framework as shown in Fig.6.1.

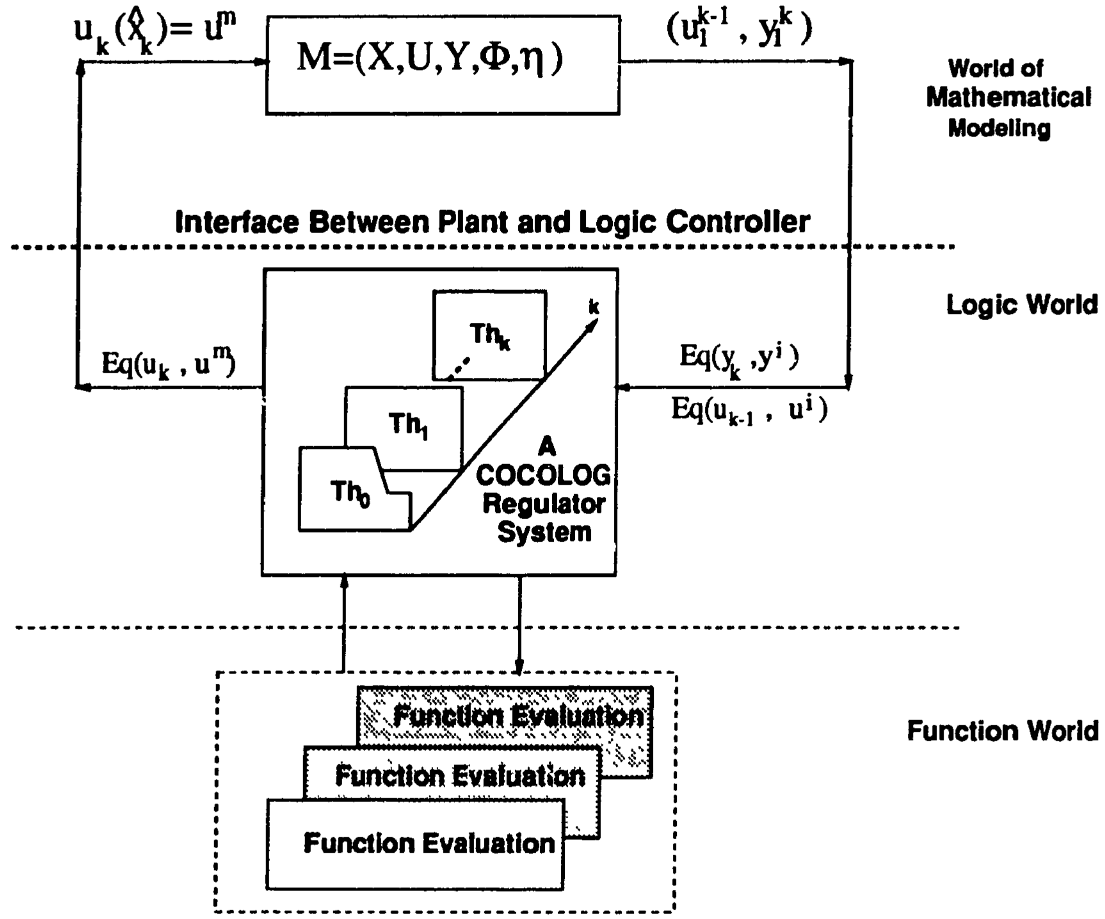


Figure 6.1: FE-Resolution for COCOLOG Theories

Where each evaluable function symbol f in L will have a corresponding evaluation function. The interface between logic-based dynamical system, i.e., logic controller and the set of evaluation functions will map each evaluable function symbol to the corresponding evaluation function with corresponding constants.

6.4 FE-Resolution and Other Attachment Techniques

In this section, we compare FE-resolution with previous attachment techniques the reader is referred to [BM81, Rus85, Gre91] for procedural attachment, [Wey77, Wey80] for semantic attachment, [Sti85] for theory resolution, and the recent work by Myers [Mye91] for universal attachment. Myers [Mye91] has compared universal attachment with previous attachment techniques and with theory resolution. Here we shall concentrate on the comparisons of universal attachment with FE-resolution.

It is known that procedural attachment is the simplest attachment method and yet is also a powerful technique linking logical function and relation symbols to programs We referred the reader to [BM81, Rus85, Gre91] for a technical presentation of the notion of procedural attachment.

Semantic attachment provides a more explicit attachment to data than procedural attachment by linking the constant symbols to data structures and function or predicate symbols to programs. Procedural attachment and semantic attachment are not motivated by the idea of removing the corresponding logical descriptions (axioms) as a benefit result of the addition of evaluation programs. It has been shown [Mye90] that both procedural attachment and semantic attachment are inherently incomplete.

Theory resolution and universal attachment are both complete Theory resolution shares the viewpoint that unsatisfiability is defined relative to given models or restricted by a given theory. Theory resolution evaluates formulas by checking the restricted unsatisfiabilities while FE-resolution evaluates a function to a value Further, theory resolution is restrictive because the test of unsatisfiability greatly increases the search space. This does not happen to the FE-resolution since unsatisfiability is checked by the search of the empty clause via resolution and function evaluation.

FE-resolution is motivated by (i) the fact that many control problems can be expressed in terms of functions and (ii) the fact that of logical treatment of function symbols is highly inefficient due to the creation of infinite Herbrand universe. Universal attachment is an extension of previous attachment techniques. It puts emphasis on the evaluation of ground or quantified formulas and provides truth values for the attachable formulas, not every formula is attachable since not every formula is evaluable, to the logic language. The possible truth value for a quantified formula can be either *true, false* or *unknown*. The shared variable between an attachable literal and a subformula makes the attachment process unable to be completely independent of the subformula. Therefore this dependency will further limit the attachability. These issues are addressed in [Mye91]. The reason for this dependency is the fact that universal quantification does not distribute over disjunction (of literals within a clause). For example, suppose $\forall x P(f(x))$ is evaluated to be false, in that case $\forall x(P(f(x)) \vee Q(x))$ does not permit us to get $Q(x)$, since $\forall x(P(f(x)) \vee Q(x)) \not\equiv \forall x P(x) \vee \forall x Q(x)$. On the other hand, in FE-resolution, we may allow the shared variable case to be processed. We evaluating at the function level, i.e., if $b \equiv f(a)$ for some a then we can resolve $P(b) \vee Q(a)$. Obviously, $P(f(x)) \vee Q(x) \models P(b) \vee Q(a)$ and $P(f(x)) \vee Q(x) \not\equiv P(b) \vee Q(a)$. But we claim, theoretically, multiple (probably infinite if necessary) number of inference steps will make the inference step to be logically equivalent, i.e., $P(f(x)) \vee Q(x) \equiv \{P(b_i) \vee Q(a_j) : i, j \in I\}$. In most cases, this logically equivalence is not necessary. Moreover, we have shown that this FE-resolution is complete in Section 6.3.

Complexity reduction analysis and experiments on automatic theorem proving in COCOLOG by the use of FE-resolution are currently under study.

Chapter 7

Conclusion

7.1 Main Results and Contributions

In the following we shall summarize our main results and contributions in the order of their apparences in this thesis.

Constructing Observers

In Chapter 2 we presented modeling of discrete state, discrete time and finite dimensional systems by input-state-output finite machines. Then we obtained the procedures to construct a current or an initial state observer. The complexity of the size of such observers measured by the number of states and input/output spaces are given in terms of initial or current state observability dags. Our results[1] about the complexity of the size of the observability dags are summarized in the Table 7.1:

Constructing Controllers

We presented procedures to construct controllers which take output from observers and generate controls for the next steps to steer the state estimate to a target state. By

[1]We assume X, U and Y denote the set of the states, controls and outputs, in the base finite machine $\mathcal{M} = (X, U, Y, \Phi, \eta)$ respectively

Upper Bound	$\|U\| = 1$		$\|U\| = 2^{\|\eta^{-1}(y_i)\|}$	
	initial observer	current observer	initial observer	current observer
Depth	$\|X\|$	$\|X\|$	$\|X\|$	$\|X\|^2/2$
Width	$\|X\|$	$\|X\|$	$O(2^{\|X\|/3})$	$O(2^{\|X\|/2})$
Size	$2\|X\| - 1$	$\|X\|^2$	$O(2^{\|X\|/3})$	$O(2^{\|X\|/2})$

Table 7.1: Size of the Observability Dags

Computational Complexity	Completely Observed	Partially Observed
Initial State Observable	$O(\|U\|\|X\|^4)$	$O(\|U\|\|Y\|\|X\|^2\|\widehat{X}\|^3)$
Current State Observable	$O(\|U\|\|X\|^5)$	$O(\|U\|\|Y\|\|X\|^3\|\widehat{X}\|^3)$

Table 7.2: Computational Complexity of Constructing Controllers

the observer and controller framework, we provide a closed loop solution to the control of systems modeled by finite machines. The dynamic programming principle has been extended to handle the partially observered control problems in the form of generating such controllers for both completely observed and partially observed initial or current state observable finite machines. The computational complexity of consturcting these cotnrollers are summarized in the Table 7.2:

Logic-based Dynamical Systems-LDS

In part II of this thesis, we presented a new paradigm to design and to implement control systems that are modeled by finite state machines. The logic-based dynamical system consist of a family of logical theories. It distinguishes itself from other logic systems by nesting a sequence (or a tree) of logics to represent the (every possible) path of the interactions of the controlled system and the environment. A specific LDS, a conditional observer and controller logics–COCOLOG is given, in Chapter 5 for the problem of designing observers and controllers for a partially observed input-state-output finite machine. A semantics is supplied for each COCOLOG theory in terms of interpretations of controlled transitions on a tree of state estimate sets. Then we obtained consistency and completeness result of COCOLOG theories. Furthermore, through the unique model property of COCOLOG theories we established the decidability though t[illegible] is not totally surprised since each COCOLOG theory is intended to characterize a snapshot of a given finite machine.

FE-Resolution

Last, in Chapter 6, we introduced an extension of resolution technique to include function evaluation capabilities. This is by replacing the logical descriptions of function symbols and hence their apparence in a deduction system with evaluated values From the construction of Herbrand universe we know that function symbols are the source of complexity. We have shown that FE-resolution is complete, in the sense that an empty clause $\Box$ is derivable from a set of clauses $\Sigma_{\mathcal{M}} \bigcup F$ by the standard resolution principle if and only if the empty clause is derivable by the FE-resolution from $\Sigma_{\mathcal{M}}$ where $\Sigma_{\mathcal{M}}$ is a set of clauses and F is a set of clauses that describe the function symbols to be replaced by the corresponding function evaluations in FE-resolution. It has been shown in Chapter 6 that the FE-resolution is complete.

7.2 Future Work

The framework and approaches presented in this thesis have opened many problems to be studied in the future. This section we just list some of the immediate questions as a direction of future works after this thesis.

Size of the Observability Dags

In Chapter 2 we have shown that size of the initial or current state observer tree can reach up to the power set of the state space. This is true when the input space of the base machine has the size of the power set of the state space of the base machine.

Furthermore we know that when the input space shrinks into a singleton set then the size of the observer tree will reduce to the square of the state space of the base machine. A challenging question will be what happens when the size of input space of the base machine is limited by a polynomial function of the size of the state space of the base machine. Or more directly what is the size limit in terms of $|X|, |U|$ and $|Y|$ if they in fact affect the size limit under concern.

Dynamic Programming and Supervisory Control Framework

We present in Chapter 3 a dynamic programming based control solution to steer state estimate sets into a desired target state. This closed loop control provides a tube of state trajectories from the current state estimate to the target state when the system state is not completely observed. It will be interesting to know that the relationships between the solutions provided by this dynamic programming formulation and provided by the supervisory control framework under partial state information.

Oracle-Functional vs Logic-based Solutions to Control Problems

For a classical controller to provide a solution to each control problem $\omega_p \in \Omega_p$, a set of feedback control laws of cardinality $card(\Omega_p)$ must be precomputed and an *oracle* must then "switch on" the appropriate controller $c_p = c(\omega_p)$. Obviously $card(\Omega_p)$ can be extremely large even for close time horizons. Consider, for instance, problems of the form: *Steer the (unobserved initial) state of $\mathcal{M}$ at k_0 into the set $X^T_{k_1}$ at k_1 (if possible), then steer the (unobserved) state in $X^T_{k_1}$ into $X^T_{k_2}$ at k_2 (if possible), then $\cdots$ at k_m (if possible).* In comparison, the logic controller does not solve each of the problems in Ω_p, but, *at the cost of the complexity of automatic theorem proving procedures in* COCOLOG, only solves the given problem $\omega^*_p \in \Omega_p$-whose complexity can be taken to be some function of the length of ω^*_p. Moreover, at an arbitrary time instant $k, k_0 \leq k \leq k_m$, the control problem ω^*_p can be changed to $\omega^{**}_p \in \Omega_p$ and the same complexity considerations outlined above still apply. A systematic comparisons of these two frameworks are considered to be an interesting and important subject for the future research.

Logic-based Adaptive Control

Logic-based dynamical system provides a declarative description to the control problem. The flexibility and expressiveness of logic language opens new dimensions to the challenging problems like adaptive control and machine learning.

The adaptive feature of the logic-based control system is apparent. Let us suppose the observer states of a observer tree $\widehat{\mathcal{M}_C}$ have been steered to the good state $\widehat{\{x_k\}}(o_1^k)$. Then suppose the dynamics of $\mathcal{M}$ changes to $\mathcal{M}'$ and the current state observer tree $\widehat{\mathcal{M}_C}$ changes to $\widehat{\mathcal{M}'_C}$. The description of this change is trivial in the logic-based system We must now recompute the observer sub-tree leading to the target state x^T and the associated (new) good states. It is evident that $\widehat{\{x_k\}}(o_1^k)$ can only be steered to x^T under the new dynamics if $\widehat{\{x_k\}}(o_1^k)$ is in the new set of good states defined for the new observer tree $\widehat{\mathcal{M}'_C}$.

Appropriate formulation of the adaptive control problem in terms of logic-based control system and comparisons between the classical formulation and logic based formulation are expected to provide more insights to the foundations of system and control science and artificial intelligence.

Complexity of FE-Resolution

The complexity of using FE-resolution to prove COCOLOG theorems is expected to be effective. A formal analysis of complexity reduction of FE-resolution over the standard theorem proving techniques, standard resolution and paramodulation in particular, is considered to be an important and challenging subject to be carried out. This also includes efficiency comparisons between function evaluation, i.e., FE-resolution and predicate evaluation, i.e., Universal Attachment.

Levesque [H.L86, H.L89] has classified the set of clauses according to their syntactic forms. The class of *vivid* clauses consist of a collection of ground, function-free atomic sentences with unique name assumption to each constant symbols and closed world assumption over domain and over each predicate, and axioms of equality. To this class of clauses, KB, it is known that $KB \models Q_1, \cdots, Q_n\alpha$ has an $O(m^{n+1}|\alpha|)$ algorithm. Further, a clause is called a Horn clause if it has the form of

$$\forall x_1, \cdots, \forall x_n, ((P_1 \bigcap \cdots, \bigcap P_k) \rightarrow P_{k+1}) \qquad n, k \geq 0, P_i \text{ is atomic}$$

For a variable-free set of Horn clauses, KB, it is known that $KB \models \alpha$ has on $O(|KB||\alpha|)$ algorithm, [WJ84]. It is interesting to know the complexity of proving theorems in COCOLOG with FE-resolution since a theorem in COCOLOG can be expressed generally as a clause with variables and functions.

Bibliography

[Ber87] Bertsekas. *Dynamic Programming.* Prentice Hall Englewood Cliffs, NJ, 1987

[Bib90] Wolfgang Bibel. Short proofs of the pigeonhole formulas based on the connection method. *Journal of Automated Reasoning*, 6:287–297, 1990.

[BM81] R.S. Boyer and J.S. Moore. *Proving them corret and Using them efficiently as new proof procedures,The Correctness Problem in Computer Sceınce.* Academic Press, Inc., New York, 1981.

[Cai88] Peter E. Caines. *Linear Stochastic Systems.* John Wiley and Sons, Inc., New York, 1988.

[CGW91] P.E. Caines, R. Greiner, and S. Wang. Classical and Logic-Based Dynamic Observers for Finite Automata. *IMA Journal of Mathematical Control & Information*, 8:45–80, 1991.

[CL73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.

[CW89a] Peter E. Caines and Suning Wang. Classical and Logic-Based Regulators Design and Its Complexity. In *Proceeding of the 28th IEEE Conference on Decision and Control*, Tampa, Florida, December 1989.

[CW89b] Peter E. Caines and Suning Wang. Classical and Logic-Based Regulators for Partially Observed Automata: Dynamic Programming Formulation. In *Proceeding of the 1989 Conference on Information Sciences and Systems*, John Hopkins University, Baltimore, MA, March 1989.

[CWG88] Peter E Caines, Suning Wang, and Russell Greiner. Dynamical Logic Observers for Finite Automata. In *Proceeding of the 1988 Conference on Information Sciences and Systems*, pages 50–56, Princeton University, Princeton NJ, March 1988.

[Dij74] E.W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communication of ACM*, 17(11), November 1974.

[End72] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, Inc., New York, 1972.

[GK87] I. Gertner and R. Kurshan Logical analysis of digital circuits. preprint, AT&T Bell Laboratories, Murray Hill, NJ, 1987.

[GN87] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[Gol87] Robert Goldblatt. *Logics of Time and Computation*. CSLI/Stanford, Stanford, CA, 1987.

[Gre91] C.C. Green. Application of theorem proving to problem solving. In *In the Proceeding of the First International Joint Conference on Artificial Intelligence*, 1991.

[Hak85] H. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[H.L86] H.Levesque. Making believers out of computers. *Artificial Intelligence*, 30, 1986.

[H.L89] H.Levesque. Logic and the complexity. Krr-tr-89-2, University of Toronto, January 1989.

[Ho87] Y.C. Ho. Performance evaluation and perturbation analysis of discrete event dynamic systems. *IEEE Trans. on Automatic Control*, July 1987.

[Hoa85] C. Hoare. *Communicating Sequential Processes* McGraw-Hill, 1985

[Kow79] Robert Kowalski. *Logic for Problem Solving* North-Holland, New York, 1979

[Kur86] R. Kurshan. Testing of containment of ω-regular languages preprint, AT&T Bell Laboratories, Murray Hill, NJ, 1986.

[Laf88] S. Lafortune. Modelling and analysis of transaction execution in database systems. *IEEE Trans. on Automatic Control*, 33(5), May 1988.

[Llo84] John Wylie Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, German, 1984.

[LW85] S. Lafortune and E. Wong. A state model for the concurrency control problem in data base management systems. Memo ucb/erl m85/27, Electroni Systems Laboratory, College of Engineering, University of California Berkeley, CA, April 1985.

[LW87] Feng Lin and Murray W. Wonham. On Observability of Discrete-event Systems Sytems Control Group Report 8701, Department of Electrical Engineering, University of Toronto, Jan 1987.

[MD80] D. McDermott and J. Doyle. Nonmonotonic Logic I. *AIJ*, 13 41–72, 1980.

[Men64] Elliott Mendelson. *Introduction to Mathematical Logic.* Van Nostrand Reinhold Company, New York, N.Y. 10001, 1964.

[MM79] G. Milne and R. Milner. Concurrent processes and their syntax. *Journal of ACM*, 26, 1979.

[Moo85] R.C. Moore. Semantical Considerations on Nonmonotonic Logic. *AIJ*, 25:75–94, 1985.

[MP81] Z. Manna and A. Pnueli. Verification of Concurrent Programming, Part 1: The Temporal Framework. *The Correctness Problem in Computer Science*, 1981. edited by S. Boyer and J.S. Moore, International Lecture Series in Computer Science.

[Mye90] K.L. Myers. Automatically generating universal attachments through compilation. In *In Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990.

[Mye91] K.L. Myers. Universal attatchment:an intergration method for logic hybrids. In *In the Proceeding of the Conference on Knowledge Representation 1991*, 1991.

[Oku88] K. Okumura. Protocol analysis from language structure. preprint, IBM Research, Tokyo Research Laboratory, 1988.

[Ost87] Jonathan S. Ostroff. *Real-Time Computer Control of Discrete Systems Modeled by Extended Machines A Temporal Logic Approach*. PhD thesis, University of Toronto, Jan 1987.

[Ost89] Jonathan S Ostroff. Synthesis of Controllers for Real-time Discrete Event Sytems. Technical Report CS-98-09, Department of Computer Science, York University, Jun 1989.

[OW85] J. Ostroff and M. Wonham. A temporal logic approach to real time control In *In the Proceeding of 24th IEEE Conference on Decision and Control*, 1985.

[OW89a] Jonathan S. Ostroff and Murray W. Wonham A Framework for Real-time Discrete Event Control. Systems Control Group Report 9810, Department of Electrical Engineering, University of Toronto, July 1989

[OW89b] C.M. Ozveren and A.S. Willsky. Tracking and restrictability in discrete event dynamic systems. Lids-p-1894, Massachusetts Institute of Technology, July 1989.

[Ozv89] C.M. Ozveren. *Analysis and Control of Discrete Event dynamic systems A state space approach.* PhD thesis, Massachusetts Institute of Technology, Aug 1989.

[Pet81] J.L. Peterson. *Petri net Theory and the modeling of systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[Pnu79] A. Pnueli. The temporal semantics of concurrent programs *Semantics of Concurrent Computation*, 1979 Lecture Notes in Computer Science 70

[RCV88] A. Fawaz R. Cieslak, C. Desclux and P. Varaiya Supervisory control of discrete-event processes with partial observations. *IEEE Trans on Automatic Control*, March 1988.

[Rei80] Raymond Reiter. A Logic for Default Reasoning *AIJ*, 13(1,2) 81–132, APR 1980.

[RK87] Stanley J. Rosenschein and Leslie Pack Kaelbling. The Synthesis of Digital Machines with Provable Epistemic Properties Technical Note 412, SRI Inter-ational, April 1987.

[Rob65] J.A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

[Rob68] J. A. Robinson. The generalized resolution principle. *Machine Intelligence*, 3:77–94, 1968.

[Rus85] Stuart Russell. *The Complete Guide to MRS*, JUN 1985. Stanford KSL Report HPP-85-12.

[RW69] G. A. [illegible]obinson and L. Wos. Paramodulation and theorem proving in first order theories with equality. *Machine Intelligence*, 4:135–150, 1969.

[RW87] P. Ramadge and M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM J. of Contr. Optimization,*, 25(1), Jan 1987.

[RW89] Peter J. Ramadge and Murray W. Wonham. The Control of Discrete Event Systems. In *Proc. IEEE, Vol.77, No.1*, pages 81–98, January 1989.

[Sha78] A.C. Shaw. Software descriptions with flow expressions. *IEEE Trans. Software Engineering.,*, 4(3), 1978

[Sti85] M E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1(4), 1985.

[TW86] J.G. Thistle and M. Wonham. Control problems in a temporal logic framework. *International Journal of Control*, 44(4), 1986.

[Wey77] R.W. Weyhrauch. *A User Mannual for FOL*, 1977. Stanford University, Technical Report STAN-CS-77-432.

[Wey80] R.W. Weyhrauch. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, 13, 1980.

[WJ84] W.Dowling and J.Gallier. Linear-time algorithms for testing the satisfiability of propositionsal horn formulae. *Journal of Logic Programming*, 3, 1984

[WR87] M. Wonham and P. Ramadge. On the supremal controllable sublanguage of given language. *SIAM J. of Contr. Optimization,*, 25(3), May 1987.

[ZW88] H. Zhong and M. Wonham. On hierachical control of discrete event systems In *In the Proceeding of 22nd Conference on Information Sciences and Sytems*, 1988.