Proceedings of the 27th Conference
on Decision and Control
Austin, Texas • December 1988

DYNAMICAL LOGIC OBSERVERS FOR FINITE AUTOMATA

P.E. Caines * R. Greiner † S. Wang ‡

* Dept. of Elect. Eng., McGill Univ., 3480 University St., Montreal, P.Q. H3A 2A7 Canada
and *Canadian Institute for Advanced Research*
Tel.(514)-398-7129, em:peterc@moe.mcrcim.mcgill.edu

† Dept. of Comp. Scie., Univ. of Toronto, 10 Kings College Rd. Ont., M5S 1A4, Canada
Tel.(416)-978-6277, em: greiner@ai.toronto.edu

‡ Dept. of Elect. Eng. McGill Univ., 3480 University St., Montreal, P.Q. H3A 2A7 Canada
Tel.(514)-398-5982, em:wsn@moe.mcrcim.mcgill.edu

Abstract

A state estimation problem is formulated for a partially observed input-state-output (I-S-O) automaton and the concept of a dynamical (default) logic observer is introduced. A simple illustrative example is then presented in which a classical dynamical observer and a dynamical logic observer are constructed to solve the observation problem for a partially observed DFA.

Key Words— discrete event system, automata, dynamical logical system, observability, default logic and artificial intelligence

1. INTRODUCTION

Artificial Intelligence (AI) and *Systems and Control Theory* (SCT) have evolved using utterly distinct mathematical methods: on the one hand significant parts of AI are based upon mathematical logic, and on the other SCT is founded upon the mathematics of controlled input-output or input-state-output dynamical systems (stochastic or deterministic). Furthermore, questions of semantics (i.e. the determination of interpretations, or models, for the formulas of a logical or a formal system) play a role in AI, and in particular in the theory of *Knowledge Representation* (KR), see [Rosenchein, 1985], while such issues play no visible role in the conventional mathematical framework used in SCT.

Despite this apparently vast gulf between the current methodology and foundational issues of AI and SCT, the objectives of both subjects, described in the broadest terms, are remarkably similar, namely, the effective use of information to reason about, and thence to control, a given system or environment.

The purpose of this paper is to show that AI and SCT have an intersection (or product!) containing a set of problems that possess the conceptual features of both subjects. Pioneering steps in this direction have been taken in [Ramadge and Wonham, 1984], [Rosenchein, 1985], [Rosenchein and Kaebling, 1987], Wonham, 1985] and [Wonham and Ramadge, 1984]. Here, however, we take a different approach from those authors. To be specific, we shall take simple class of dynamical systems represented by partially observed automata and then pose the state estimation problem in terms of (i) the problem of constructing a *classical dynamical system* (CDS) which generates a sequence of state estimates, and (ii) the problem of the construction of a *dynamical logic system* (DLS) which generates a sequence of propositions that correctly describe properties of the state of the automaton. In particular, we are interested in those cases where the classical dynamical observer system *estimates* converge to the correct values of the system state and the dynamical logic system *statements* converge (in an appropriate sense) to true characterizations of the system state. When such convergent observer systems exist we shall call the automaton *observable* or *logically observable* respectively. (We shall make this statement technically precise later in the paper.) The development of this theory requires *inter alia* the definition of dynamical logic systems, and this is perhaps one of the main contributions of the paper. The final section of this paper contains an illustrative example applying these ideas to an extremely simple, but non-trivial, partially observed automaton.

A further remark concerning motivation is appropriate. Apart from its interest as an intrinsic property of dynamical systems, observability is important in systems and control theory because of its role in the analysis of regulators (or supervisors). Generally speaking, output-to-input or state feedback regulators may be designed to steer the state of a partially observed system to a required value if and only if the system is both controllable and observable. (For a presentation of these ideas in the context of linear systems, see Kalman, Falb and Arbib, 1969] [Caines, 1988]). It is this that motivates us to examine *logic observability* and *logic controllability* (i.e. controllability by use of dynamical logic systems) for automata in order that the problem of what we shall term *logic regulations* of automata may be addressed and analysed.

2. THE DYNAMICAL OBSERVER PROBLEM FOR FINITE AUTOMATA

In this paper, an automaton is taken to mean a deterministic state output finite automaton. The dynamical observers for the finite automata discussed in this section are themselves modeled by finite automata.

Definition 2.1 *An input-state-output (I-S-O) (finite) automaton* is a quintuple $M = (X, U, Y, \Phi, \eta)$ where
X is a (finite) set of *states*,
U is a (finite) set of *inputs*,
Y is a (finite) set of *outputs*,
$\Phi : X \times U \to X$ is a *transition function*,
$\eta : X \to Y$ is a *output function*. □

As notations used in this paper, we sometimes may write $\Phi(x, u)$ as $\Phi_u(x)$, u_i^n as the sequence $u_i u_{i+1} u_{i-2} \cdots u_n$, where u_j denotes the input at the instance $j \in \mathbf{Z}_-$, and the same for y_i^n.

The dynamical evolution of an input-state-output automaton $M = (X, U, \Phi, \eta, Y)$ can be displayed by taking U^* to be the set of all finite sequence of inputs and by extending $\Phi : X \times U \to X$ to be $\Phi : X \times U^* \to X$, where for $\forall i, n \in \mathbf{Z}_-, \forall u_i^n \in U^*$ and $\forall x \in X, \Phi$ has the property that the following equation is satisfied:

$$\Phi(x, u_i^n) = \Phi(\Phi(x, u_i), u_{i+1}^n)$$

Because η is not necessarily a one-to-one map, an I-S-O automaton will often be referred to as a *partially observed automaton*.

The input-state-output automaton set-up described above includes, as a special case, that of any conventional deterministic finite automaton A with partial transition function

88CH2531-2/88/0000-0226$1.00 © 1988 IEEE

δ. (In the standard case where no outputs are defined for A we set $Y = \phi$, to obtain the appropriate restriction of the general I-S-O automaton set-up). Whenever δ is defined, set $\Phi(x,u) = \delta(x,u)$, and whenever it is not adjoin to M a state x_J (indicating jam) and define Φ to be such that $\Phi(x,u) = x_J$. Further postulate $\Phi(x_J,u) = x_J$ for all $u \in U$ (i.e. the system can never get out of a jam). Finally the conventional automata theoretic notion of an *event* is recovered by defining the event map $e : X \times U \rightarrow U \bigcup \{e_J\}$, taking values in the event set $U \bigcup \{e_J\}$, to be such that any transition of the system M of the form $x_{k+1} = \Phi(x_k,u_k)$, with $x_{k+1} \neq x_J$, is mapped into an event $e_k = e(x_k,u_k) \triangleq u_k$. Further, the distinguished event e_J and the map e are assumed to satisfy

$$e_k = e_J, \text{ whenever } \Phi(x_k,u_k) = x_J$$

In other words, whenever the system jams the event sequence is terminated with an (unbounded) sequence e_J, e_J, ... This has the formal language interpretation that an event sequence, or a word. which corresponds to a sequence of state transitions ending in the jammed state generates a word $e_1, e_2, \ldots\ldots, e_k, e_J, e_J \ldots\ldots$ which is declared to be ungrammatical, i.e. does not lie in the generated language $L(A)$.

The I-S-O automaton can also obviously cover the situation where the conventional automata states are absorbed into a terminal state corresponding to the termination of a grammatical or admissible string in the generated language $L(A)$.

The observer problem for an automaton $M = (X,U,Y, \Phi,\eta)$ is the problem of determining the initial or current state of the automaton from observations on its inputs and outpus over a finite time period.

Let an initial state for a input-state-output automaton M be denoted x_1^* and let an input-state-output sequence of length N starting with the initial state x_1^* be given by

$$\begin{aligned}(u_1^{N-1}, x_1^{*N}, y_1^N) &= (u_1^{N-1}, x_1^{*N}, \{\eta(x_i^*); 1 \leq i \leq N\}) \\ &= (u_1^{N-1}, \{\Phi(x_1^*, u_1^{i-1}); 1 \leq i \leq N\}, \\ &\quad \{\eta(\Phi(x_1^*, u_1^{i-1})); 1 \leq i \leq N\})\end{aligned}$$

Definition 2.2 An *N-consistent state sequence* with respect to the output sequence $y_1^M = \{\eta(x_i^*); 1 \leq i \leq M\}$, denoted $[x_1^N\widehat{]}_M$, is any state sequence x_1^N that satisfies

$$\begin{aligned}x_{k-1} &= \Phi(x_1^*, u_1^k); 1 \leq k \leq N-1 \\ \eta(x_k) &= y_k, 1 \leq k \leq M\end{aligned}$$

□

In other words. an N-consistent state sequence with respect to the output sequence y_1^M is one that both satisfies the system dynamics up to N and gives the observed output sequence up to M.

Definition 2.3 A *state sequence estimate* with respect to an output sequence y_1^M of a state sequence of length N denoted $\{x_1^N\widehat{\}}_M$ is defined to be the union of all N-consistent state sequences with respect to y_1^M. In symbols:

$$\{x_1^N\widehat{\}}_M \triangleq \bigcup [x_1^N\widehat{]}_M \triangleq (\bigcup ([x_1^N\widehat{]}_M)_1, \cdots, \bigcup ([x_1^N\widehat{]}_M)_N) \tag{2.1}$$

where $([x_1^N\widehat{]}_M)_i$ denotes the i_{th} component vector of any one of the N-consistent state sequences $[x_1^N\widehat{]}_M$. □

Definition 2.4. A *state estimate (at M)* with respect to y_1^N, denoted $\{\widehat{x_M}\}_N$ is the set of M_{th} elements of a state sequence estimate with respect to y_1^N, i.e.

$$\{\widehat{x_M}\}_N \equiv P_M(\{x_1^N\widehat{\}}_N) = \bigcup ([x_1^N\widehat{]}_N)_M \tag{2.2}$$

where P_M is the projection operator and $P_M(Z_1, \cdots Z_{M-1}, Z_M, Z_{M+1}, \cdots) = Z_M$ and $Z_i \subset X$ □

Two particularly important special cases are: (i) the *initial state estimate* with respect to y_1^N and $\{x_1^N\widehat{\}}_N$ defined by $\{\widehat{x_1}\}_N \triangleq P_1(\{x_1^N\widehat{\}}_N)$, and (ii) the *current state estimate* with respect to y_1^N, defined by $\{\widehat{x_N}\}_N \triangleq P_N(\{x_1^N\widehat{\}}_N)$.

Example we can illustrate these definitions with the state-output automaton given in Fig -2.1-: Let $x_1^* = x1$ and so let the automaton generate the following state-output sequences with initial segments

$N = 1$
$(x_1^* = x1, 1)$
$N = 2$
$((x_1^* = x1, x_2^* = x3), (1,0))$
$N = 3$
$((x_1^* = x1, x_2^* = x3, x_3^* = x1), (1,0,1))$

In these cases we have

$N = 1$
$\{x_1^1\widehat{\}}_1 = (\{x1, x2\})$,
and hence
$\{\widehat{x_1}\}_1 = \{x1, x2\}$;
$N = 2$
$\{x_1^2\widehat{\}}_2 = (\{x1\}, \{x3\})$,
and hence
$\{\widehat{x_1}\}_2 = \{x1\}$ and $\{\widehat{x_2}\}_2 = \{x3\}$;
and finally $N = 3$
$\{x_1^3\widehat{\}}_3 = (\{x1\}, \{x3\}, \{x1\})$
and so
$\{\widehat{x_1}\}_3 = \{x1\}, \{\widehat{x_2}\}_3 = \{x3\}$ and
$\{\widehat{x_3}\}_3 = \{x1\}$.

We can formulate the following definitions:

Definition 2.5. An input-state-output automaton $M = (X,U,Y, \Phi,\eta)$ is said to be *initial state observable* iff $\forall x^* \in X, \forall u \in U^*, \exists K \in \mathbf{Z}_+, s.t. \forall N \geq K$

$$\{\widehat{x_1}\}_N \equiv P_1(\{x_1^N\widehat{\}}_N) = x^* \qquad \square$$

Definition 2.6. An input-state-output automaton $M = (X,U,Y, \Phi,\eta)$ is said to be *current state observable* iff $\forall x^* \in X, \forall u \in U^*, \exists K \in \mathbf{Z}_+$. s.t. $\forall N \geq K$

$$\{\widehat{x_N}\}_N = \Phi(x^*, u_1^{N-1}) \,\overline{\nabla}\, x_N^* \qquad \square$$

Clearly, as illustrated by the following state-output automaton, that the initial state observability implies current state observability but not vice versa.

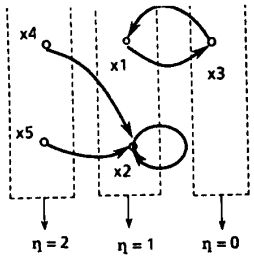

Fig.-2.1-

Definition 2.7. Two states $x_1', x_1'' \in X$ of an automaton $M = (X, U, Y, \Phi, \eta)$ are said to be *(observability) equivalent*, denoted by $x_1' \equiv_o x_1''$ iff $\forall u \in U^*, \eta(\Phi(x_1', u)) = \eta(\Phi(x_1'', u))$ □

The equivalence relation $\equiv_o$ over the state space X of an automaton $M = (X, U, Y, \Phi, \eta)$ induces a partition over the equivalence classes of $X/\equiv_o$ which permits us to define another property for automata.

Definition 2.8. An automaton $M = (X, U, \Phi, \eta, Y)$ is said to be in *reduced form* iff

$$\forall x \in X, [x]_{\equiv_o} = \{x\}. \qquad \square$$

Clearly an I-S-O automaton cannot be initial state observable if there exist observability equivalent states. Further, a reduced state automaton can be seen to be initial state observable.

We can now give some useful formulae for the initial and current state estimate sets. The first is a simple set intersection expression whereas the second has the important *predictor-corrector* form of many recursive algorithms in systems and control theory. In fact, in this case, we should perhaps refer to the recursion as a *predictor-refiner* formula, since no error-correction in the usual sense of the words take place.

Proposition 2.1 Consider any I-S-O automaton, then for all initial states $x1 \in X$ and any input $u \in U^*$ the following equations hold:

$$\widehat{\{x_1\}}_N = \bigcap_{k=1}^{N} \Phi^{-1}_{u_1^{k-1}}(\eta^{-1}(\eta(\Phi_{u_1^{k-1}}(x1)))) \tag{2.3}$$

$$\widehat{\{x_N\}}_N = \bigcap_{k=1}^{N} \Phi_{u_k^{N-1}}(\eta^{-1}(\eta(\Phi_{u_1^{k-1}}(x1)))) \tag{2.4}$$

and the predictor-refiner recursion formula

$$\widehat{\{x_1\}}_N = \widehat{\{x_1\}}_{N-1} \bigcap \Phi^{-1}_{u_1^{N-1}}(\eta^{-1}(\eta(\Phi_{u_1^{N-1}}(x1)))) \tag{2.5}$$

$$\widehat{\{x_{N-1}\}}_{N-1} = \Phi_{u_N}(\widehat{\{x_N\}}_N) \bigcap \eta^{-1}(\eta(\Phi_{u_1^N}(x1))) \tag{2.6}$$

where $\widehat{\{x_N\}}_0 \triangleq X$. and $\Phi^{-1}_{u_1^k}(A)$ for $A \subset X$ is defined to be the subset of X such that $\forall z \in \Phi^{-1}_{u_1^k}(A), \Phi_{u_1^k}(z) \subset A$. □

All the proofs of the Propositions, Corollarys and Theorems that appear in this paper will be omitted for the limitation of the space.

Corollary 2.1 For any I-S-O automaton $M = (X, U, \Phi, \eta, Y)$, M is *initial state observable* iff $\forall x_1 \in X, \forall u \in U^*, \exists K \in \mathbf{Z}_+, s.t. \forall N \geq K$

$$\begin{aligned}\widehat{\{x_1\}}_N &= \bigcap_{k=1}^{N} \Phi^{-1}_{u_1^{k-1}}(\eta^{-1}(\eta(\Phi_{u_1 k-1}(x1)))) \\ &= \widehat{\{x_1\}}_{N-1} \bigcap \Phi^{-1}_{u_1^{N-1}}(\eta^{-1}(\eta(\Phi_{u_1^{N-1}}(x1)))) \\ &= \{x1\}\end{aligned} \tag{2.11}$$

and M is *current state observable* iff $\forall x_1 \in X, \forall u \in U^*, \exists K \in \mathbf{Z}_+, s.t. \forall N \geq K$

$$\begin{aligned}\widehat{\{x_N\}}_N &= \bigcap_{k=1}^{N} \Phi_{u_1^{N-1}}(\eta^{-1}(\eta(\Phi_{u_1^{k-1}}(x1)))) \\ &= \Phi_{u_{N-1}}(\{\widehat{x_{N-1}}\}_{N-1}) \bigcap \eta^{-1}(\eta(\Phi_{u_1^N}(x1))) \\ &= \Phi_{u_1^{N-1}}(x1)\end{aligned} \tag{2.12}$$

□

Corollary 2.2 For any S-O automaton $M = (X, Y, \Phi, \eta)$ the following equations hold:

$$\begin{aligned}&\Phi^{N-1}(\bigcap_{k=1}^{N} \Phi^{-(k-1)}(\eta^{-1}(\eta(\Phi^{k-1}(x_1))))) \\ &= \bigcap_{k=1}^{N} \Phi^{N-1}(\Phi^{-(k-1)}(\eta^{-1}(\eta(\Phi^{k-1}(x_1))))) \\ &= \bigcap_{k=1}^{N} \Phi^{N-k}(\eta^{-1}(\eta(\Phi^{k-1}(x_1))))\end{aligned} \tag{2.13}$$

□

Corollary 2.3 For any S-O automaton $M = (X, Y, \Phi, \eta)$ the following equation holds:

$$\begin{aligned}&\Phi(\bigcap_{k=1}^{N} \Phi^{N-k}(\eta^{-1}(\eta(\Phi^{k-1}(x))))) \\ &= \bigcap_{k=1}^{N} \Phi(\Phi^{N-k}(\eta^{-1}(\eta(\Phi^{k-1}(x)))))\end{aligned} \tag{2.14}$$

□

Proposition 2.2 An I-S-O automaton $M = (X, U, Y, \Phi, \eta)$, M is *initial state observable* if and only if the following implication holds: $\forall x_1', x_1'' \in X, \forall u \in U^*, \exists K \in \mathbf{Z}_+$, s.t. $\forall N \geq K$

$$\begin{aligned}&\eta(\Phi(x_1', u_1^k)) = \eta(\Phi(x_1'', u_1^k)), 1 \leq k \leq N \\ &\Longrightarrow x_1' = x_1''\end{aligned} \tag{2.15}$$

and M is *current state observable* if and only if $\forall x_1', x_1'' \in X, \forall u \in U^*, \exists K \in \mathbf{Z}_+$ such that $\forall N \geq K$

$$\begin{aligned}&\eta(\Phi(x_1', u_1^k)) = \eta(\Phi(x_1'', u_1^k)), 1 \leq k \leq K \\ &\Longrightarrow \Phi(x_1', u_1^{N-1}) = \Phi(x_1'', u_1^{N-1})\end{aligned} \tag{2.16}$$

□

Definition 2.9. Given an I-S-O automaton $M = (X, U, Y, \Phi, \eta)$, a *(classical) dynamical observer system* of M is an automaton $\widehat{M} = (\widehat{X}, \widehat{U}, \widehat{\Phi}, \widehat{\eta}, \widehat{Y})$ s.t. $\widehat{U} = Y, \widehat{Y} = X$ and $\widehat{\eta} : \widehat{X} \to X$. □

$\widehat{M}$ takes the output space Y of M as its input space and it prints out its "estimate" of the state of the automaton M.

Definition 2.10. A dynamical observer system $\widehat{M}$ is said to be *convergent* in finite time if $\forall x1 \in X, \exists K \in \mathbf{Z}_+$, s.t. $\forall N \geq K$ and $\forall u_1^N \in U^*$ we have $\widehat{\eta}(\widehat{\Phi}(\widehat{x1}, \widehat{u}_1^N)) = \Phi(x1, u_1^N)$. □

3. DYNAMICAL LOGIC SYSTEMS

To reason about and hence to control a dynamical system. i.e. an input-(state)-output system, we need to choose a suitable formal language or a suitable logic which must be able to support such concepts as *system input and output, system state. feedback, time instant* and those concepts associated with the dynamical evolution of these quantities. Moreover, this reasoning must be carried out in (discrete) *real* or *system* time. 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.

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

We take L to denote a set of *well formed formulas* (wffs), ς 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 explore a necessary and sufficient condition for this definition by considering the equivalence classes over the set of all possible axioms as follows.

Let $\vdash$ be a relation and $\varsigma \vdash \omega$ denote the wff ω is derivable from the set of axioms ς. $\mathrm{Th}(\varsigma) = \{\omega : \omega \in L, \varsigma \vdash \omega\}$ is the set of all theorems (or the range of the deductive closure operator) derivable in L under ς. Clearly, 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 $\varsigma_1, \varsigma_2 \in 2^L$ by $\varsigma_1 \equiv_{\mathrm{Th}} \varsigma_2$ if and only if $\mathrm{Th}(\varsigma_1) = \mathrm{Th}(\varsigma_2)$. Furthermore, the equivalence class of ς induced by $\equiv_{\mathrm{Th}}$ will be denoted by $[\varsigma]_{\mathrm{Th}}$. Hence we can define a time invariant logic as follows:

Definition 3.1. A logic L is said to be *time invariant* w.r.t. system time iff the set of axioms ς of L is invariant w.r.t. system time up to the equivalence class $[\varsigma]_{\mathrm{Th}}$. L will be said to be a *time varying* logic if otherwise. □

Most classical and contemporary logics are time invariant 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 Petri-Net, Communicating Sequential Processes. Even the so called Temporal Logic [Manna and Pnueli, 1981] 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 (or a theorem) is not derivable at a given instant will not be derivable at all system time clock instants and hence no *error correction* is permitted or possible.

In contrast to this, in AI, the logics that are used by (some) knowledge based systems can update their rules of inference, and facts in the data base, upon receiving signals from operators or from some learning schemes. These are actually the time-varying logics or dynamical logics (*not* dynamic logics) in the sense we discussed in this paper. Within AI, these issues are being addressed formally by the construction of formal systems such as default logic see [Reiter, 1980], or non-monotonic logics in general, see [McDermott and Doyle, 1980] and [Moore, 1985], which are intended to be formulations of common sense reasoning or reasoning under uncertainty. (K. Konolige [Konolige, 1987] proved that a default logic is equivalent to an autoepistemic logic, one type of non-monotonic logic introduced by R.C. Moore see [Moore, 1985]).

Dynamical Logic Systems

The general idea of a dynamical logic system (DLS), is inspired by that of a dynamical system, that is to say a non-anticipative mapping ξ from the input (time) function space $\mathcal{U}$ to the output (time) function space $\mathcal{Y}$. Before we present this idea in detail, let us further consider the relationship between the system time and logic time in a general logic system.

As we introduced earlier, the logic time instant will take place between each system time clock period, and each logic time unit will correspond to each logic inferential step in the logic deduction process. Hence corresponding to three possible assumptions on the deduction machine, we have three different models on the relations between the system time and logic time, which are discussed as follows:

(i). $t = 1$. — The deduction machine is extremely slow, it can take only one inferential step during each system time period.

(ii). $t = 2^M$ for some $M \in \mathbf{Z}_+$. — This implies that the deduction machine is relatively fast, there exists a rather large number of logic time units between two system time clock instants, but still may not be able to reach the deductive closure. This assumption is interesting in the sense it represents most practical systems.

(iii). $t = \infty$. — This is an ideal case, for which the deductive closure can always be achieved.

Depending upon the different deduction machines used by a DLS, we get different system mappings ξ.

Suppose we are working on a resource limited agent, namely, there exists an upper bound on the number of logic time units that can be taken between any succeeding system time clock instants. In this case, we can write out the mapping ξ as PTh, a *partial theorem operator*, which stands for the set of theorems derived during each system time clock period. Let PTh_t denote the partial set of theorems, i.e. the subset of the deductive closure derived by L_ξ up to time t. Notice that, this set of theorems may not even include the set of new axioms that are received as the current input, i.e. u_t may not lie in PTh_t. Clearly, we have the following equation:

$$\mathrm{PTh}_{t+1} = \mathrm{PTh}(\mathrm{PTh}_t, u_{t+1}) \tag{1}$$

We leave the evaluation of the PTh operator to be unfixed in order to model different real world situations.

In an extreme case, if we assume we are dealing with an ideal agent i.e. the deduction machine can take infinitely many logical inferential steps during each system time period, then we can identify the system mapping ξ as the deductive closure operator Th, as mentioned before.

$$\begin{aligned}
[\xi(u)]_{t+1} &\triangleq \mathrm{Th}(u_0^{t+1}) \\
&\overline{\nabla}\ \mathrm{Th}_{t+1} \\
&= \mathrm{Th}(u_0^t \cup \{u_{t+1}\}) \\
&= \mathrm{Th}(u_0^t, u_{t+1}) \\
&= \mathrm{Th}(\mathrm{Th}(u_0^t), u_{t+1});\ \text{since } \mathrm{Th}(u_0^t) = \mathrm{Th}(\mathrm{Th}(u_0^t)) \\
&= \mathrm{Th}(\mathrm{Th}_t, u_{t+1}) \qquad (2')
\end{aligned}$$

If we use x_t to replace PTh_t in (1) or Th_t in (2') then a state equation formula is obtained from:

$$\begin{aligned}
x_{t+1} &= \mathrm{PTH}(x_t, u_{t+1}) \\
&\text{or} \qquad (2'') \\
x_{t+1} &= \mathrm{Th}(x_t, u_{t+1})
\end{aligned}$$

Hence PTH_t or Th_t play a role of a state in a DLS. We can further introduce a concept of a *logic state of a dynamical logic system* as follows.

Definition 3.2.— A *logic state (minimal logic state)* x_t, at the (system) time t, of a dynamical logic system L_ξ is a *(minimal)* set of theorems of L_ξ, which, together with the input function $U(t')$ will uniquely determine the output function $Y(t')$ of the system for $\forall t' \geq t$. □

As for a classical dynamical system, we may obtain a logic state for an input-output dynamical logic system by constructing the Nerode equivalence classes (see [Caines, 1988], [Kalman, Falb, Arbib, 1969]) over the set of all possible inputs, i.e. the input space. Therefor, any input-output DLS will give rise to an input-state-output DLS.

So far the output function $Y(t)$ in our DLS has been formulated as the set of all theorems derived up to t. i.e. PTh_t or Th_t depends on whether the deduction machine used by DLS is an ideal one or not. In another words, the output is taken

to be the state value of the DLS, but in general, we may define the output function $Y(t)$ to be a subset of PTh_t or Th_t. We can formalize this idea by the following concepts.

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. A *base level query map* of $Q(t)$ to a DLS at the time t is defined to be a time dependent set $Y(t)$ such that $Y(t) = \{\omega \in L : \omega \in \mathrm{Th}_t$ and $\omega \in Q(t)\}$, (or $Y(t) = \{\omega \in L : \omega \in \mathrm{PTh}_t$ 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 DLS.

Definition 3.3.—An *output function* $Y(t)$ of a DLS L_ξ is a base level query map of $Q(t)$ to L_ξ. □

Definition 3.4.— A *dynamical logic system* (DLS) consists of a septuple (FR(L), LA, DyA, DaA, Linf, Dinf, Q), indexed by time, given by a set of formation rules FR(L), axioms (LA, DyA, DaA) and rules of inference (Linf, Dinf) such that the logic state recursion is given by (2) and the output map is given by the query map Q. □

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

(1.) Formation Rules
(2.) Axioms
  (2.1) Logical Axioms
  (2.2) Dynamical (Environment) Axioms
  (2.3) Data Axioms
(3.) Inference Rules
  (3.1) Logical Inference Rules
  (3.2) Dynamical Inference Rules
(4.) Query map

The formulation of the concept of a dynamical logic system (DLS) 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 3.5.—An *input-output dynamical logic system* L_ξ of the time varying logic L_ξ is a triple $L_\xi = (\mathcal{U}, \mathcal{Y}, \xi)$ where

$$\mathcal{U} \triangleq \{U : \mathbf{Z}_- \to 2^L\} \equiv \text{ input space}$$
$$\mathcal{Y} \triangleq \{Y : \mathbf{Z}_- \to 2^L\} \equiv \text{ output space}$$
$$\xi : u_0^t = (u_0, u_1, \cdots u_t) \longmapsto y_0^t = (y_0, y_1, \cdots y_t)$$

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 following schema.

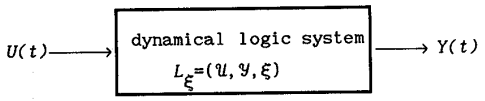

Fig.-3.1-

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.

We use $[\xi(\cdot)]_t$ to denote the evaluation of a mapping ξ at the time t. A *shift operator* is therefore defined by $[S_\tau(\xi(\cdot))]_t = \xi(\cdot)_{|t-\tau}$, which represents the evaluation of the mapping ξ is shifted backward τ units of time clock instants. Here we can define the notion of a time invariant input-output dynamical logic system as:

Definition 3.6.— An (input-output) dynamical logic system L is said to be *time invariant* iff

$$[\xi(U(\cdot))]_t = [S_{-\tau}(\xi(S_\tau(U(\cdot))))]_t \qquad \square$$

We observe that by the construction of a DLS, it yields a time invariant (input-output) dynamical logic system. Further, if we define P_t be a (nonanticipative) truncation operator, by

$$[P_t U(\cdot)]_\tau = \begin{cases} U(\tau) & \text{if } t \geq \tau \\ \phi & \text{otherwise} \end{cases}$$

then we can define a *nonanticipative (input-output) dynamical logic system* L_ξ by:

Definition 3.7.— An (input-output) dynamical logic system L_ξ is said to be *nonanticipative* iff for $\forall U_1, U_2 \in \mathcal{U}$ $P_t U_1 = P_t U_2$ implies $P_t(\xi(U_1)) = P_t(\xi(U_2))$ □

By Data Axioms 2.3 we see that any DLS is necessarily nonanticipative. For a general discussion of dynamical systems see [Kalman, Falb and Arbib, 1969] and [Caines, 1988].

In the rest of this paper we will formulate a state observation problem for a partially observed DFA in terms of the dynamical logic system.

DYNAMICAL (DEFAULT) LOGIC OBSERVERS FOR FINITE AUTOMATA

The state estimation problem of a finite automaton will give rise to a dynamical logic system, in the sense as we discussed above. Where the new observations about the system will introduce new data axioms for the DLS and the partial observation will lead us to adopt a suitable set of defaults. Notice that, If we restrict the dynamical inference rules in the axiomatic system of a DLS to a set of defaults and keep the dynamical axioms in the same system be constant, then the DLS will become a sequence of default logic systems.

The concepts that associate with a dynamical logic observer (DLO) for a partially observed I-S-O automaton will be given by the following definitions.

Definition 3.8.— A *dynamical logic observer*-DLO for a partially observed I-S-O automaton, denoted as M, is a DLS for which the inputs are observations on the output sequences generated by M, the axioms of DLO contain the state transitions of M, and the outputs are base level queries on the state of M. □

The *dynamical default logic observer* (DDLO) will be a DLO where the *default rules* are adopted in the inference rules.

Definition 3.9.—A DLO for a partially observed automaton M is said to be *convergent to M in finite time*, if $\exists k \in \mathbf{Z}_-$, s.t. $\forall t \geq k$ the axioms of the DLO at the time instant k, denoted as ς_k, and the sequence of statements or predicates $ST(x_t, t)$ for $t \geq k$ which describe the state trajectory $x(t) = x_t$ of M satisfies the relation $\varsigma_k \vdash ST(x_t, t)$. □

The finite time convergent property simply says that after a finite period of time interval the theorems derived by the DLO will give true characterizations of the properties of M.

Definition 3.10.— A partially observed automaton M is said to be *logically observable* if there exists a finite time convergent DLO for M. □

A DLO works as a state predictor which receives the outputs from a DFA at each time clock instant and then (instantaneously) print out its reasoning result about state information of a given DFA. The reasoning process is a logical deduction process which maps its inputs to a 'optimal' estimate according to the best of its 'knowledge'. A complete example will be given in section 5 and comparisons will be made between a classical dynamical observer (CDO) and a dynamical default logic observer (DDLO).

4. MAIN THEOREM AND A BOUND ON THE OBSERVER AUTOMATON

In this section, we present our main theorem which connects the notion of observability of an I-S-O automaton with the existence of a *convergent classical dynamical observer* and the existence of a *convergent dynamical logic observer*. Then we describe a general design procedure of a CDO for an observable S-O automaton by introducing the notion of a *DAG observer tree* through a simple example. Finally, an upper bound on the size of a DAG observer tree for an observable S-O automaton is determined by the Theorem 4.2.

Theorem 4.1 Let $M = (X, U, Y, \Phi, \eta)$ be an input-state-output automaton, then the following statements are equivalent:

(a) M is current (respectively initial) state observable.

(b) There exists a convergent CDO for the current (respectively initial) state value.

(c) There exists a convergent DLO generating estimates of the current (respectively initial) state.

□

Consider the following initial (also current) state observable S-O automaton.

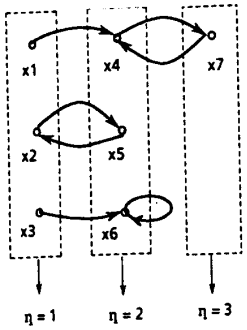

Fig.-4.1-

General procedures for designing an initial and a current state observers for the above automaton are describted by the following DAG initial and current state observer trees given in Fig.-4.2- and -4.3- respectively.

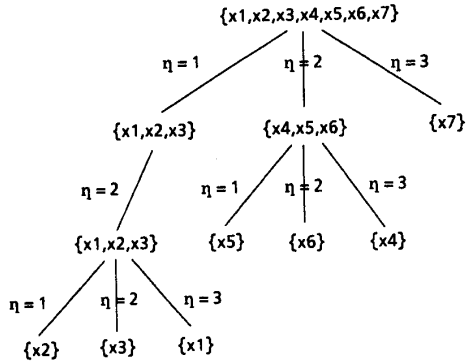

Fig.-4.2- DAG initial state observer tree

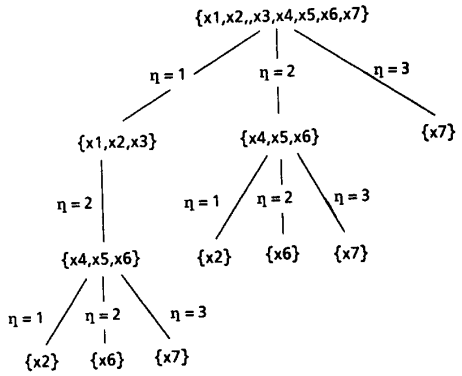

Fig.-4.3- DAG current state observer tree

The generating of the DAG observer trees in either case are self-descriptive as shown in the above Figs. and hence we omit further discussion here. The following theorem will explore the structural property as well as a bound on the size of the DAG observer trees.

Theorem 4.2 Let M be an initial state observable state-output automaton with N states and let OM be the DAG initial state observer for M. Then we have:

(a) there is a node splitting (for some non-singleton node) at each level of OM.

(b) OM has O(N) states and O(N) transitions.

(c) there exits a current state observer for M with O(N) states.

5. AN EXAMPLE

In this section we take a simple but non-trivial partially observed automaton to address the state estimation problem and to illustrate the ideas presented earlier in the paper.

The automaton to be discussed is given as $M = (X, U, Y, \Phi, \eta)$ where $X = \{x1, x2, x3\}, U = \{\alpha, \beta, \gamma\}, Y = \{1, 2\}$, $\eta : X \to Y$ and $\eta(x1) = 1, \eta(x2) = \eta(x3) = 2$. $\Phi : X \times U \to X$ is defined by $\Phi(x1, \alpha) = x2, \Phi(x2, \beta) = x1, \Phi(x3, \gamma) = x3$, other transitions are given implicitly which are not interested here. The observation problem is to predicate (estimate) the system state base on the knowledge of the system dynamics. The observation schema is shown as Fig. -5.1-

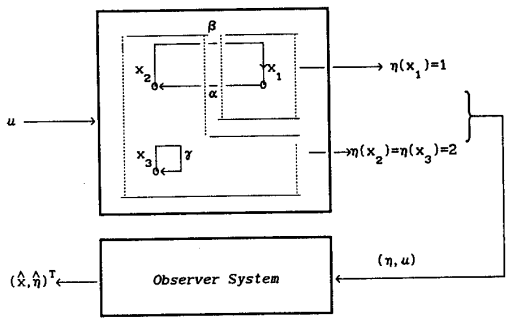

Fig-5.1-

In the following, two different observer systems, namely a CDO and a DLO, for the given automaton M will be presented and the convergent property of the observer systems will be examined.

5.1 A Classical Dynamical Observer (CDO)

A classical dynamical observer $\widehat{M} = (\widehat{X}, Y, \widehat{\Phi}, \widehat{\eta}, \widehat{Y})$ for the given automaton $M = (X, U, \Phi, \eta, Y)$ is constructed based on the dynamics of the automaton M. The state transition function $\widehat{\Phi} : \widehat{X} \times Y \rightarrow \widehat{X}$ and the output function $\widehat{\eta} : \widehat{X} \rightarrow \widehat{Y}$ is described by the Fig.-5.2-

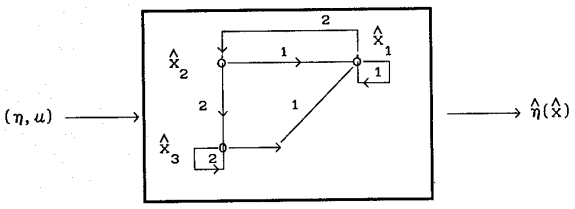

Fig.-5.2-

Claim: $\forall(x1, \widehat{x1}) \epsilon X \times \widehat{X} \quad \exists k \epsilon \mathbf{Z}_+$, s.t. $\forall t \geq k, \hat{x}_t = x_t$
Proving this convergent property by verifying all the state trajectories induced by all the initial state pairs $(x1, \widehat{x1}) \epsilon X \times \widehat{X}$. Two of the nine possible trajectories are displayed in Fig. -5.3a- and -5.3b-.

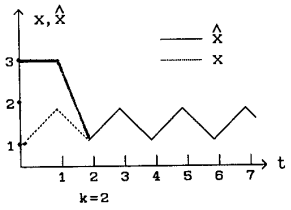

Fig.-5.3a-

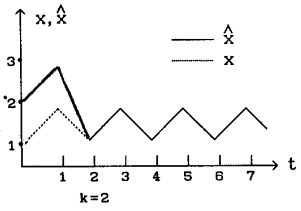

Fig.-5.3b-

5.2 A Dynamical Default Logic Observer (DDLO)

In the following a dynamical default logic observer, which is based upon a first order default logic, for the given automaton $M = (X, U, Y, \Phi, \eta)$ is constructed by the description of the axiomatic system.

1. **Symbols**
   - 1.1 Constants: $1, 2, x1, x2, x3$
   - 1.2 Variable:
     - $x, \hat{x} \epsilon \{x1, x2, x3\}$. denote state and estimated state
     - $\eta, \hat{\eta} \epsilon \{1, 2\}$. denote output and estimated output
     - $t1, t2, \epsilon \mathbf{Z}_-$, denote logic time.
   - 1.3 Predicates:
     - succ $(t1, t2)$, $t2$ is the successor of $t1$
     - $O(t, \eta)$. M output η at time t
     - EO $(t, \hat{\eta})$. $\hat{\eta}$ is the estimate output at t
     - S (t, x). M is in state x at t
     - ES $(t, \hat{x})$, $\hat{x}$ is the estimate state at t
   - 1.4 Connectives: $\neg, V, \Lambda, \rightarrow, \longleftrightarrow, =, (,)$
2. **Formulation Rules**: As in the standard first order logic.
3. **Axioms**:
   - Logical Axioms
     - A1.1. $\mathcal{A} \rightarrow (\mathcal{B} \rightarrow \mathcal{A})$
     - A1.2. $(\mathcal{A} \rightarrow (\mathcal{B} \rightarrow \mathcal{C})) \rightarrow ((\mathcal{A} \rightarrow \mathcal{B}) \rightarrow (\mathcal{A} \rightarrow \mathcal{C}))$
     - A1.3. $(\neg\mathcal{B} \rightarrow \neg\mathcal{A}) \rightarrow ((\neg\mathcal{B} \rightarrow \mathcal{A}) \rightarrow \mathcal{B})$
     - A1.4. $\forall x \mathcal{A}(x) \rightarrow \mathcal{A}(t)$
     - A1.5. $\forall x(\mathcal{A} \rightarrow \mathcal{B}) \rightarrow (\mathcal{A} \rightarrow \forall x \mathcal{B})$
   - Dynamical Axioms
     - A2.1. $O(t, 1) \rightarrow \mathrm{ES}(t, x1)$
     - A2.2. $O(t, 2) \rightarrow \mathrm{ES}(t, x2)\ V\ \mathrm{ES}(t, x3)$
     - A2.3. Succ $(t1, t2) \rightarrow (\mathrm{ES}(t1, x1) \leftrightarrow \mathrm{ES}(t2, x2))$
     - A2.4. Succ $(t1, t2) \rightarrow (\mathrm{ES}(t1, x2) \leftrightarrow \mathrm{ES}(t2, x1))$
     - A2.5. Succ $(t1, t2) \rightarrow (\mathrm{ES}(t1, x3) \leftrightarrow \mathrm{ES}(t2, x3))$
     - A2.6. $\mathrm{ES}(t, x1) \rightarrow \mathrm{ES}(t, 1)$
     - A2.7. ES $(t, x2)\ V\ \mathrm{ES}(t, x3) \rightarrow \mathrm{EO}(t, 2)$
   - Belief axioms
     - A3.1. EO $(t, \hat{\eta}) \leftrightarrow 0(t, \hat{\eta})$
     - A3.2. ES $(t, \hat{x}) \leftrightarrow S(t, \hat{x})$
   - Unique name assumption.
     - A4.1. $\neg(x1 = x2)$
     - A4.2. $\neg(x2 = x3)$
     - A4.3. $\neg(x3 = x1)$
     - A4.4. $\neg(1 = 2)$
   - Fact
     - A5.1. Succ $(1, 2)$
   - Function property
     - A6.1. $\neg\mathrm{ES}(t, x)\ V\ \neg\mathrm{ES}(t, \hat{x})\ V\ (x = \hat{x})$
     - A6.2. $\neg\mathrm{EO}(t, \eta)\ V\ \neg\mathrm{EO}(t, \hat{\eta})\ V\ (\eta = \hat{\eta})$
4. **Rules of Inference**
   - R4.1. $\frac{\alpha, \alpha \rightarrow \beta}{\beta}$ (Modus Ponens)
5. **Default Rule**
   - D5.1. $\frac{O(t.2) : M[\mathrm{ES}(t, x2)]}{\mathrm{ES}(t, x2)}$ (Default) □

We can prove this dynamical logic observer converge to a true characterization of the given automaton state in finite time by simulating the automaton starts from any of the states in $\{x1, x2. x3\}$.

Case-1.—at the time $t_s = 1$, the automaton is at the state $x1$, i.e. $S(1, x1)$
Proof (of convergence)
C1. O(1, 1): by *observation* at the $t_s = 1$
C2. ES(1. $x1$): C1 and A2.1.
By the completeness of the first order theory and the complete description of the system dynamics given from A2.1 to A2.7. everything from then on will be correct. □

Case-2.—at the time $t_s = 1$, the automaton is at the state $x2$. i.e. $S(1. x2)$
Proof (of convergence)
C1. O(1. 2); by *observation* at the $t_s = 1$
C2. ES(1. $x2$) V ES(1, $x3$); C1, A2.2
B3. ES (1. $x2$); by default proof $P_{\mathrm{ES}(1,x2)} = (\{D5.1\}. \{\})$
: see Note 1
B3.1. ES (2. $x1$): A5.1, B3, A2.4 (state estimation)
B3.2. EO (2, 1): B3.1. A2.6 (output estimation)
C4. O(2. 1): by observation at the $t_s = 2$
C5. ES (2. $x1$): C4, A2.1
C6. ES (1. $x2$): A5.1, C5, A2.4 (B3 has been confirmed)
By the same argument as in Case 1. the logical observer will give correct estimations from then on.

Case-3.—at the time $t_s = 1$, the automaton is at the state $x3$. i.e. $S(1. x3)$

Proof (of convergence)

C1. O(1,2): by *observation* at the $t_s = 1$
C2. ES(1.$x2$) V ES(1,$x3$); C1, A2.2
B3. ES (1,$x2$); by default proof $P_{\mathrm{ES(1,x2)}} = (\{D5.1\}, \{\})$
; see Note 1
B3.1. ES (2,$x1$); A5.1, B3, A2.4 *(state prediction)*
B3.2. EO (2,1); B3.1, A2.6 *(output prediction)*
C4. O(2,2); by observation at the $t_s = 2$
C5. $\neg$ EO(2,1) V $\neg$ EO (2,2); A4.4, A6.2
B3.3 $\neg$ EO (2,2); B3.2, C5
C6. EO (2,2); A3.1, C4
C7. $\neg$ ES (1,$x2$); inconsistency of B3.3 and C6. force a
; theorem revision procedure being called
; and results in C7. see Note 2.
C8. ES (1,$x3$); C2, C7

By the completeness of the first order theory and the complete description of the system dynamics given from A2.1 to A2.7, everything from then on will be correct. □

Note 1. The default proof $P_{\mathrm{ES(1,x2)}}$ w.r.t. the default theory (D,W) at the current system time is given by a top-down linear resolution and refutation process. For a formal discussion see [Reiter, 1980].

Note 2. Upon the occurrence of inconsistency, a theory revision procedure is called which will try to make minimum modifications over the current belief set B, where in this case $B = \{B3\}$ a singleton, so B3 must be eliminated, therefore the only possible solution we can get is that $\neg$ ES (1,$x2$) should be recognized. For a formal discussion on default logic see [Reiter, 1980].

6. CONCLUSION

In this paper we have introduced the notion of a DLS and the introduction contains general remarks on their significance. Here we shall confine ourselves to comments on the flexibility of dynamical logic observers. In contrast to the design of a classical observer which has fixed dynamics once the design has been completed . a dynamical logic observer is a DLS designed to produce state estimates for any system whose dynamics have been specified in the dynamical axioms (DyA) of the DLS. We note that in the dynamical logic observer set-up adaptation appears to be facilitated because the alteration of an automaton A to A$'$ requires only the alteration of DyA to DyA$'$. On the other hand, a classical observer must be totally redesinged (a priori) in response to any change in the observed system. while a general theory of the effectiveness of classical dynamical observers is readily available, at least in the linear case.

ACKNOWLEDGEMENTS

Conversations with M. Makkai and R. Reiter on the fundamentals of logic and default logic respectively are gratefully acknowledged.

REFERENCES

Caines P.E., *Linear Stochastic Systems*, John Wiley and Sons, Inc. 1988.

Dietterich G. Thomas. Learning at the Knowledge Level, *Machine Learning* Vol.1, No.3, 1986

Enderton H.B., *A Mathematical Introduction to Logic* Academic Press NYC 1972.

Goldblatt R., *Logic of Time and Computation*, Center for Study of Language and Information, 1987.

Greiner Russell, Principles of Inference Processes. *CSRI 193* University of Toronto, Jan. 1987

Kalman R.E., Falb P.L. and Arbib M.A., *Topics in Mathematical System Theory*, McGraw-Hill 1969

Konolige K., On The Relation Between Default Theories And *Autoepiste mic* Logic, In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1987, pp394-401.

Lin F. and Wonham W.M. On Observability of Discrete-Event Systems, Systems Control Group Report 8701, Department of Electrical Engineering, University of Toronto Jan. 1987

Manna Z., and Pnueli A., Verification of concurrent programs: the temporal framework, in *The Correctness Problem in Computer Science*. R.S. Boyer and J.S. Moore (eds.),Academic Press, 1981.

McDermott D. and Doyle J., Non-monotonic Logic I, *Artificial Intelligence* 13 (1980), pp41-72

Moore R.C., Semantical Considerations on Nonmonotonic Logic *Artificial Intelligence* 25 (1985), pp75-94

Newell Allen, The Knowledge Level, *Artificial Intelligence* Vol.2, 1981, pp1–20

Ramadge P.J. and Wonham., Supervisory Control of a Class of Discrete-event Processes, *Proc. Sixth Internat. Conf. Analysis and Optimization of Systems*, Nice, France, June 1984.

Reiter R., A Logic for Default Reasoning, *Artificial Intelligence* 13 (1980), pp81-132.

Rosenchein S.J., Formal Theories of Knowledge in AI and Robotics, *Technical Note* 362, 1985, Artificial Intelligence Center, SRI International, Menlo Park, California 94025.

Rosenchein S.J. and Kaelbling L.P., The Synthesis of digital Machines with Provable Epistemic Properties, SRI International and CSLI Stanford, 1987.

Wonham W.M., On the Control of Discrete-event Systems, *Presented at the Seventh International Sympo'sium on the Mathematical Theory of Networks and Systems (MTNS)*, Stockholm, June, 1985.

Wonham W.M., Towards an Abstract Internal Model Principle.*IEEE Transaction ON Systems,Man, and Cybernetics* Vol. SMC-6, No 11, Nov. 1976

Wonham W.M. and Ramadge P.J., On the Supremal Controllable Sublanguage of a Given Language, *Proc. 23rd IEEE Conf. on Decision and Control*, Las Vegas, Nevada, December 1984.