Xiaojun Yang, M.D. Lemmon and P. J. Antsaklis, "Inductive Inference of Optimal Controllers for Uncertain Logical Discrete Event Systems," P roc o f t he 1 0th I nternational S ymposium o n I ntelligent Control , pp. 585-590, Monterey, CA, August 27-29, 1995.

Inductive Inference of Optimal Controllers for Uncertain Logical Discrete Event Systems

Xiaojun Yang, Mike Lemmon [1] and Panos Antsaklis
Dept. of Electrical Engineering
University of Notre Dame, Notre Dame, IN 46556

Abstract

This paper summarizes preliminary results on the implementation and convergence of an inductive learning algorithm used to identify optimal logical DES controllers. The algorithm is a modification of Angluin's L^* procedure. It uses queries to an equivalence oracle and incomplete membership oracle to learn a minimal deterministic finite automaton consistent with the supremal controllable sublanguage of a known specification language and unknown plant. This paper shows how the proposed algorithm is implemented, provides results on the procedure's convergence properties, and presents an example illustrating its use.

1. Introduction

Discrete event system (DES) controller synthesis methods as proposed by Ramadge and Wonham [1] require that the state transition model of the plant's legal behaviours be known. It has been pointed out that this requirement is often unreasonable. In many cases, the DES plant may only be partially specified and the "optimal" DES controller must then be synthesized from on-line observations of the controlled plant's behaviour. Prior work along this line of inquiry will be found in [2], [3], and [4]. In [2], control actions at each execution step are based on an N-step ahead prediction of the controlled system's behaviour. In [3], an on-line method was proposed for resolving uncertainty in the DES plant's transitions. In [4], a modification of Angluin's L^*-procedure was proposed for identifying DES controllers. Under certain circumstances, the learning algorithm proposed in [4] appears to identify a DES controller containing the supremal controllable sublanguage. This paper examines the proposed algorithm and summarizes recent results concerning the learning procedure's convergence.

[1] The partial financial support of the National Science Foundation (MSS92-16559) and the Electric Power Research Institute (RP8030-06) are gratefully acknowledged.

The remainder of this paper is organized as follows. Section 2 reviews the L^*-algorithm. Section 3 discusses the special oracles used by the modified L^*-procedure. Section 4 presents results on the algorithm's convergence. Section 5 uses two simple examples to illustrates the procedure's use.

2. L^*-Learning of DES Controllers

Inductive inference is a machine learning procedure which is used to infer the minimal boolean functional (also called the target functional) consistent with a set of input-output pairs of that function. These input-output pairs constitute a training set from which the target functional is to be inferred. Two different types of teachers are common in many inductive learning protocols. These "teachers" evaluate the training examples and use them to contruct an approximation for the target functional. One type of teacher is called the *membership oracle*. This oracle is an algorithm which declares whether or not a given training example is a member of the target set. The second type of teacher is called an *equivalence oracle*. This oracle is an algorithm which declares whether a given boolean functional is equivalent to the target functional. In the event that they are not equivalent, then the equivalence oracle returns a *counter-example* illustrating where the two structures differ..

There are a large number of results on the feasibility of using membership and equivalence queries to identify unknown regular languages. These results are of special interest to this paper, since logical DES generate regular languages. In particular, it has been shown that the inference of minimal DFA's through the sole use of membership queries is NP-complete [5]. This results suggests that on-line DES identification is impractical [6]. The L^*-procedure proposed by Angluin in [7] provides a way out of this dilemma. This learning algorithm uses both membership and equivalence queries and was shown to have a polynomial complexity in the size of the minimal DFA and the length of the counter-examples returned by

0-7803-2722-5/95 $4.00 © 1995 IEEE

Xiaojun Yang, M.D. Lemmon and P. J. Antsaklis, "Inductive Inference of Optimal Controllers for Uncertain Logical Discrete Event Systems," P roc o f t he 1 0th I nternational S ymposium o n I ntelligent Control , pp. 585-590, Monterey, CA, August 27-29, 1995.

the equivalence oracle. Based on these theoretical results, it was conjectured in [4] that some modification of the L^*-procedure could provide an effficient means of identifying logical DES controllers.

A high level flowchart of the original L^*-algorithm is shown in figure 1. This chart can also be used to describe the overall structure of the modified L^*-procedure described in [4]. The primary difference between the modified and original L^*-procedure rests with the oracle implementations. These implementation differences will be discussed in more detail below. Before discussing the oracles, however, it will be necessary to review the algorithm illustrated in figure 1.

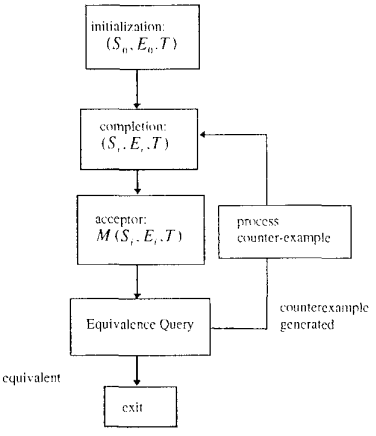

Figure 1: High level flowchart of L^* procedure

The algorithm uses membership and equivalence queries to build up an *observation table* representing the regular language to be learned. The observation table is represented by the ordered triple, (S, E, T), where S and E are prefix closed and suffix closed languages, respectively. S is sometimes referred to as a language of *accessing* strings and E is referred to as a language of *distinguishing* strings. $T : (S \cup S\Sigma)E \rightarrow \{0, 1\}$ is a partial function which agrees with the declarations of the membership oracle over strings in $(S \cup S\Sigma)E$. The observation table can be represented in tabular form as shown below. In this table, $\Sigma = \{a, b, A, B\}$, $S = \{\varepsilon, a, aA\}$, and $E = \{\varepsilon, A\}$. ε is the null event. The rows are labeled with string in S and $S\Sigma - S$. The columns are labeled with strings in E. An entry in the table is therefore indexed by a string $s \in S \cup S\Sigma$ and a string $e \in E$. An entry in the table is the membership oracle's evaluation of the string se. A value of 1 indicates that the string se is a member of the target functional and 0 indicates otherwise.

		ε	A
S	ε	1	1
	aA	1	0
	a	1	1
$S\Sigma - S$	aB	1	1
	aa	1	1
	aAB	1	0
	aAb	1	1
	B	1	1
	A	1	0

Define a function $row : (S \cup S\Sigma) \rightarrow \{0, 1\}^{|E|}$ which maps elements $se \in S \cup S\Sigma$ onto a string of length $|E|$. This function returns a single row of the table shown above. A language is said to be consistent with the table if and only if all strings, se, in the table such that $T(se) = 1$ are members of the language. The rows of the table can be used to identify right invariant equivalence classes of a regular language consistent with the table. This is accomplished by *completing* the table. A table is said to be closed if for all $s\sigma \in S\Sigma$ there exists an $t \in S$ such that $row(s\sigma) = row(t)$. A table is said to be consistent if for any s and t in S such that $row(s) = row(t)$, then for all $\sigma \in \Sigma$, $row(s\sigma) = row(t\sigma)$. A table that is closed and consistent is said to be complete. Any observation table can be closed and completed through a simple algorithmic procedure (see [7] for details). The table shown above is a completed table. We can define an equivalence relation R_L such that sR_Lt if and only if $row(s) = row(t)$. Because the table is completed, R_L is right invariant and can be used by the MyHill-Nerode characterization [8] to construct a DFA accepting a regular language consistent with the table. In particular, it has been shown [7] that any closed and consistent table will have an automaton which is minimal in the sense that any other automaton consistent with the table will have more states. For the observation table shown above, the associated DFA is shown in figure 2. The label for the states in this figure are the distinct rows of the observation table.

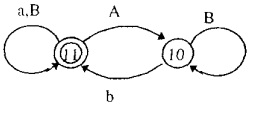

Figure 2: Observation Table's Acceptor

The computational procedure illustrated in figure 1 constructs a completed observation table in the following manner. Given an initial set S_0 and E_0 (usually chosen to be the null event, ε), the procedure evaluates the observation table, (S_0, E_0, T). It then

Xiaojun Yang, M.D. Lemmon and P. J. Antsaklis, "Inductive Inference of Optimal Controllers for Uncertain Logical Discrete Event Systems," P roc o f t he 1 0th I nternational S ymposium o n I ntelligent Control , pp. 585-590, Monterey, CA, August 27-29, 1995.

completes the table to obtain (S_1, E_1, T) and constructs an acceptor, $M_1 = M(S_1, E_1, T)$. This acceptor is then presented to the counter-example oracle as an hypothesized DFA for the target language, K. If M_1 is not equivalent to the acceptor for K, then the oracle returns a *counter-example*, s. The counter-example is either accepted by M_1 and not in K, or else it is an element of K which is not accepted by M_1. Such counter-examples are generally constructed by search methods. The counter-example and its prefixes are then added to S and the algorithm loops back to complete the table again. This iterative process is then repeated until the counter-example oracle returns no more counter-examples. It has been shown [7] that each counter-example adds at least one more state to the acceptor. This means that the L^*-procedure constructs a sequence of acceptors with a monotone increasing number of states. Since the target language is assumed to be a finite language, this immediately implies the algorithm's convergence to the minimal DFA representing the target language.

3. Membership and Equivalence Oracles

To use the L^*-algorithm to learn logical DES controllers an appropriate membership and equivalence oracle need to be defined. Assume the plant language, $L(G)$, and the control specification, $\bar{K}$, are regular prefix-closed languages over an event alphabet, Σ. It is assumed that Σ is partitioned into controllable, Σ_c, and uncontrollable Σ_u events. The plant is assumed to be observable. The specification and the uncontrollable events are assumed to be known, but the plant language is unknown. Recall that a language S is controllable if and only if $\bar{S}\Sigma_u \cap L(G) \subseteq \bar{S}$. The supremal controllable sublanguage $K^\uparrow$ of the specification $\bar{K}$ is the largest controllable sublanguage of $\bar{K}$. The objective is to compute $K^\uparrow$ from on-line observations of the plant's behaviours. Based on our preceding discussion, it certainly appears to be reasonable to use the L^*-algorithm to identify $K^\uparrow$. Direct use of the L^*-algorithm, however, requires a membership oracle declaring whether or not a given string s is in $K^\uparrow$. A plant behaviour, s, is said to be legal with respect to $\bar{K}$ if $s \in \bar{K}$. Clearly an illegal (w.r.t $\bar{K}$) behaviour is not in $K^\uparrow$. A legal behaviour, however, will be in $K^\uparrow$ if and only if there are no uncontrollable suffixes in $L(G)$ taking this legal string outside of $\bar{K}$. This determination cannot be made without knowledge of $L(G)$ and so we cannot implement a membership oracle for $K^\uparrow$. The significance of the preceding discussion is that a modification of the L^*-procedure will need to be developed if we are to use it in identifying logical DES controllers.

The modifications to the L^*-algorithm involve two changes. First, we will introduce a conditional membership oracle which is "updated" using illegal observed plant behaviours. Second, we introduce a modified equivalence oracle whose counter-examples are processed in different manners depending upon the available information on plant behaviours and their uncontrollable suffixes. With these two modifications, the original flowchart shown in figure 1 can still be used to identify the optimal DES controller. Convergence results for this procedure are provided in section 4.

It is well known that computation of the supremal controllable sublanguage can be performed in an iterative manner using the following formula [9]

$$K_0 = \bar{K} \quad (1)$$

$$K_{i+1} = K_i - [(L(G) - K_i)/\Sigma_u]\,\Sigma^* \quad (2)$$

This iteration produces a sequence of languages by removing uncontrollable plant behaviours in $L(G) - K_i$ from the original specification. This procedure clearly produces a monontone decreasing sequence of languages which converges to the supremal controllable sublanguage after a finite number, N, of iterations;

$$K^\uparrow = K_N \subseteq K_{N-1} \subseteq \ldots \subseteq K_1 \subseteq K_0 = \bar{K} \quad (3)$$

Note that each, K_i, can be computed from the preceding one once we have an observed uncontrollable plant behaviour which is "illegal" with respect to K_{i-1}. The preceding equations therefore provide a means of including observed plant behaviours into the original specification and, in particular, we can then view this sequence of languages as a sequence of membership oracles which converge to the membership oracle for $K^\uparrow$ after a finite number of "illegal" behaviours are observed.

The preceding discussion therefore leads to our first modification of the L^*-procedure. Whereas the original L^*-procedure used a static membership oracle, we will use a membership oracle which is conditioned on a list of observed illegal plant behaviours. A plant behaviour $st \in L(G)$ will be said to be uncontrollably illegal if s is legal, $t \in \Sigma_u^* - \varepsilon$ and $st \notin \bar{K}$. Let C be a collection of observed uncontrollably illegal plant behaviours. The language formed by discarding the uncontrollable suffixes of strings in C will be denoted as

$$D_u(C) = \{\, s \in L(G) \text{ such that } st \in C \text{ and } t \in \Sigma_u^* \} \quad (4)$$

The set C can be viewed as a set of sample plant behaviours that are used to update the memberhsip oracle. Let C_i be a set of i observed uncontrollably illegal plant behaviours and let C_{i+1} be $C_i \cup \{s_{i+1}\}$

Xiaojun Yang, M.D. Lemmon and P. J. Antsaklis, "Inductive Inference of Optimal Controllers for Uncertain Logical Discrete Event Systems," P roc o f t he 1 0th I nternational S ymposium o n I ntelligent Control , pp. 585-590, Monterey, CA, August 27-29, 1995.

where s_{i+1} is an observed uncontrollably illegal behaviour. We therefore have a growing set of observed behaviours which can be used to modify the membership oracle. In particular, we now introduce a membership oracle conditioned on the set C_i and represented by the partial function T_i. This partial function is defined as follows for $i \geq 1$:

$$T_0(s) = \begin{cases} 0 & \text{if } s \notin \bar{K} \\ 1 & \text{if } s \in \bar{K} \end{cases}$$

$$T_i(s) = \begin{cases} 0 & \text{if } T_{i-1}(s) = 0 \text{ or } s \in D_u(C_i)\Sigma^* \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

The function $T_i(s)$ can then be used to evaluate and complete a given observation table in exactly the same way as done in the traditional L^*-algorithm. An immediate consequence of the preceding discussion is that the language consistent with T_i will be $\bar{K} - D_u(C_i)\Sigma^*$. This is precisely the same type of iteration shown in equation 2 and it can be expected to converge to $K^\uparrow$ after a finite number of updates.

The polynomial complexity of the L^*-algorithm arises from its use of equivalence queries. These equivalence queries provide a set of counter-examples showing where the hypothesized acceptor and the target language disagree. Once again because of uncontrollable plant behaviours, the equivalence oracle in the original L^*-procedure must be modified. In the original L^*-procedure, counter-examples occur when a legal plant behaviour s is not accepted by the observation table's current acceptor $M(S, E, T)$ or when s is an illegal behaviour generated by $M(S, E, T)$. The introduction of uncontrollable events requires us to distinguish between controllable and uncontrollable behaviours. A controllable behaviour which is illegal denotes a disagreement between the target language and the current acceptor. However, uncontrollable behaviours will be illegal because uncontrollable events led a legal string outside of $\bar{K}$. This undesirable behaviour is a consequence of the specification (membership oracle) not being controllable and the appropriate way for processing these counter-examples is to modify the membership oracle. In particular, this is precisely the type of processing implied by the conditional membership oracle shown in equation 5.

The counter-example oracle used by the modified L^*-procedure searches for four different types of counter-examples. The search can be carried out in a variety of ways. A very straightforward method is proposed below. The proposed counter-example oracle performs a randomized search through the enabled sets of observed plant behaviours. The search alternates between using the specification, K_i, or the acceptor M_i to control the plant. K_i is used to enable plant behaviours in the following manner. We first randomly generate a string in Σ^* which is legal with respect to K_i. This string is used to enable a sequence of plant events. If the resulting observed plant behaviour is not accepted by M_i or if it results in an uncontrollably illegal behaviour, then the search has found a counter-example to the equivalence query. Alternatively, the acceptor M_i can be used to enable plant behaviours. In this case, the acceptor enables controllable transitions in the plant and the resulting observed behaviour is examined. If the observed plant behaviour is illegal with respect to K_i, then the search has found another counter-example to the equivalence query.

As noted above, the counter-examples returned by the oracle can be categorized into four groups. In the following, a behaviour will be said to be unaccepted if it is not accepted by $M_i = M(S, E, T_i)$. A legal behaviour will be legal with respect the current membership oracle K_i. The four types of counter-examples are itemized below;

1. unaccepted legal plant behaviours which were generated by using K_i as the controller,
2. illegal controllable plant behaviours which were generated by using M_i as a controller,
3. illegal behaviours which were generated by using K_i as the controller,
4. and illegal uncontrollable plant behaviours which were generated by using M_i as a controller.

The first two cases are counter-examples in the usual sense treated by the L^*-procedure. These strings are processed by adding their prefixes to S and completing the resulting table with membership oracle T_i. The third and fourth cases produce counter-examples which are used to update the membership oracle. In this case, the counter-example is added to the previous set C_i to form C_{i+1} and the observation table is then re-evaluated using membership oracle T_{i+1}.

4. Convergence Results

The convergence of the proposed learning algorithm follows immediately by combining convergence results from the L^*-algorithm and the iterative computation in equation 2. The relevant results of interest are stated below.

Theorem 1 *[7] Assume that (S, E, T) is a closed, consistent observation table. Suppose the acceptor $M(S, E, T)$ has n states. If M' is any acceptor consistent with T that has n or fewer states, then M' is isomorphic to $M(S, E, T)$.*

Xiaojun Yang, M.D. Lemmon and P. J. Antsaklis, "Inductive Inference of Optimal Controllers for Uncertain Logical Discrete Event Systems," P roc o f t he 1 0th I nternational S ymposium o n I ntelligent Control , pp. 585-590, Monterey, CA, August 27-29, 1995.

Theorem 2 *Assume that the counter-example s has been added to observation table (S, E, T). Let (S', E', T') denote the updated and completed observation table. If $M(S, E, T)$ has n states, then the acceptor $M(S', E', T')$ has at least $n + 1$ states.*

Proof: By adding s to (S, E, T) we obtain an observation table (S', E', T'). This table is consistent with (S, E, T) and by theorem 1, $M(S', E', T')$ has at least n states. Since s is a counter-example whose prefixes are added to S, we know that s cannot be accepted by $M(S, E, T)$. Therefore, the new acceptor, $M(S', E', T')$ cannot be isomorphic to $M(S, E, T)$ and must therefore have at least $n + 1$ states. •

Theorem 3 *[9] If the languages $L(G)$ and $\bar{K}$ are regular, then the sequence of languages, K_i, defined in equation 2 converges to $K^\uparrow$ after a finite number of updates.*

The first result states that any acceptor $M(S, E, T)$ extracted from a complete observation table will be "minimal" in the sense that any other acceptor consistent with the table is isomorphic to $M(S, E, T)$ or has more states than $M(S, E, T)$. The second result [7] states that each counter-example adds at least one state to the minimal acceptor extracted from the updated observation table. The third result states that the sequence of membership oracles represented by the iteration in equation 2 converges to the supremal controllable sublanguage after a finite number of updates. The following theorem combines these known results to prove the convergence of the proposed logical DES identification scheme.

Theorem 4 *The modified L^* procedure with membership oracle updating converges to an acceptor, K, such that the controlled language $L(G|K)$ is the supremal controllable sublanguage, $K^\uparrow$. Furthermore this acceptor is identified after a finite number of counter-examples are detected and used to update the observation table.*

Proof: By the preceding description of counter-examples, it should be apparent that there are two distinct types of counter-examples. The first type of counter-example is used to update the membership oracle K_i, and the second type is a "true" counter-example in the sense originally used by the L^*-procedure. These counter-examples are used to update the observation table. Let the "true" counter-examples be placed in set D and let the other type of counter-example be placed in set C. The acceptor constructed from the modified L^*-procedure can be expressed as $M(S_j, E_j, T_i)$ where S_j E_j are the access and the distinguishing sets of the table that occur after the jth counter-example in D has been added to the table. Let T_i denote the membership oracle's partial function after the ith counter-example in C has been used to update it. The number of states in $M(S_j, E_j, T_i)$ be denoted as $n_{i,j}$, the minimal number of states for the acceptor of T_i will be denoted as n_i, and the number of states in the supremal controllable sublanguage's DFA is $n^\uparrow$. From theorem 3, we known that $n_i \to n^\uparrow$ after a finite number of updates. From theorem 2, we know that $n_{i,j} \leq n_i$. Therefore the number of states, $n_{i,j}$ in the sequence of acceptors is overbounded by a convergent sequence n_i. Since n_i converges after a finite number of updates and since $n_{i,j} \to n_i$ (consequence of theorems 1 and 2), we conclude that $n_{i,j} \to n^\uparrow$ after a finite number of counter-examples. •

5. Example

A simple example of the modified L^*-algorithm is provided below. In this example we consider an event alphabet $\Sigma = \{\varepsilon, a, b\}$ where b is uncontrollable and ε is the null event. The unknown plant language is $L(G) = \overline{a^*ba^*}$ and the specification language is given by $\bar{K} = \overline{a^n} \cup \overline{a^k ba^*}$ where $k \leq 2$ and $n > 2$. Note that this is an uncontrollable specification.

As noted above, the initial membership oracle is the prefix-closed specification $\bar{K}$. We construct an initial observation table by taking $S = \varepsilon$ and $E = \varepsilon$. The resulting observation table is shown below. This table is clearly complete. The acceptor extracted from this table is shown in figure 3.

	ε
ε	1
a	1
b	1

The acceptor for the observation table is then used as a controller and generates controllable and uncontrollable plant behaviours. The procedure stops searching when an illegal (w.r.t $\bar{K}$) plant behaviour is identified by the membership oracle. The simulation program written to implement this procedure produced the illegal accepted plant behaviour *aaab*. This behaviour is used to update the observation table as shown below. The acceptor extracted from this table is shown in figure 3.

Xiaojun Yang, M.D. Lemmon and P. J. Antsaklis, "Inductive Inference of Optimal Controllers for Uncertain Logical Discrete Event Systems," P roc o f t he 1 0th I nternational S ymposium o n I ntelligent Control , pp. 585-590, Monterey, CA, August 27-29, 1995.

	ε	a	aa
ε	1	1	1
a	1	1	0
aa	1	0	0
b	1	1	1
ab	1	1	1
aab	1	1	1

As before, the acceptor is used to control the plant. In this case no illegal behaviours were uncovered. We then begin using the original specification to generate controllable plant behaviours and see if there are any accepted legal behaviours. In this case an unaccepted legal plant behaviour $baaa$ is discovered and added to the observation table. The resulting completed table is shown below. The acceptor extracted from this table is shown in figure 3. This is the supremal controllable sublanguage, K^1, for the specification language.

	ε	a	aa	aaa
ε	1	1	1	0
a	1	1	0	0
b	1	1	1	1
aa	1	0	0	0
ba	1	1	1	1
baa	1	1	1	1
$baaa$	1	1	1	1
ab	1	1	1	1
aab	1	1	1	1
$baaaa$	1	1	1	1

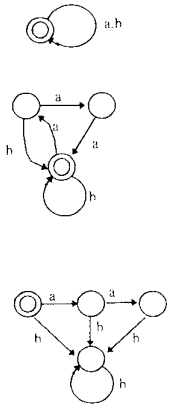

Figure 3: sequence of acceptors generated by modified L^*-procedure

6. Summary

This paper examined the behaviour of an inductive inference algorithm proposed in [4] for on-line DES controller synthesis. The proposed algorithm is a modification of Angluin's L^*-procedure which updates the membership oracle from observed uncontrollable illegal behaviours in such a way that the oracle decides the membership of strings in the supremal controllable sublanguage, K^1, of a known specification language, K. The algorithm has been shown to converge after a finite number of membership oracle updates to a minimal DFA consistent with K^1. The important open question concerns the number of updates required for convergence. The number of counter-examples will depend on the equivalence oracle used and we suspect that an appropriately formulated heuristic search procedure might be very useful in quantifying the complexity of this learning procedure.

References

[1] P. Ramadge and W.M. Wonham (1987), "Supervisory control of a class of discrete event processes", *SIAM Journal of Control and Optimization*, Vol 25, No. 1., pp. 206-230, Jan. 1987.

[2] S-L Chung, S. Lafortune, F. Lin (1992), "Limited lookahead policies in supervisory control of discrete event systems", *IEEE Trans. Automatic Control*, Vol 37, No. 12, pp. 1921-1935, Dec. 1992.

[3] S. Young, Vijay Garg (1991), "Transition uncertainty in discrete event systems", *Proceedings 6th IEEE International Symposium on Intelligent Control*, pp. 245-250, Arlington, VA, Aug. 1991.

[4] M. Lemmon, P. Antsaklis, X. Yang, C. Lucisano (1995), " "Control System Synthesis through Inductive Learning of Boolean Concepts", *IEEE Control Systems Magazine*, June 1995.

[5] D. Angluin (1978), "On the complexity of minimum inference of regular sets", *Int. J. Information and Control*, Vol. 39, pp. 337-350, 1978.

[6] J.N. Tsitsitklis (1989), "On the control of discrete-event dynamical systems", *Mathematics of control, signals, and systems*, Vol 2., No. 1, pp. 95-107, 1989.

[7] D. Angluin (1987), "Learning regular sets form queries and counter-examples", *Int. J. Information and Computation*, Vol 75, No. 1, pp. 87-106, 1987.

[8] J.E. Hopcroft and J.D. Ullman (1979), *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass., 1979.

[9] W.M. Wonham and P.J. Ramadge (1987), "on the supremal controllable sublanguage of a given language", *SIAM Journal of Control and Optimization*, vol 25, No. 3, pp. 637-659, 1987.