1994 IEEE International Symposium on
Intelligent Control
16-18 August, 1994.
Columbus, Ohio, USA

A Total Ordering on Languages with a Bipartite Alphabet

Amit Anil Nanavati * Sandeep Gulati † S. S. Iyengar ‡

Abstract

We consider here the Ramadge–Wonham framework for discrete event (dynamical) systems. Therein arise situations when the Supervisory Marking Problem (SMP) or the Supervisory Control Problem (SCP) is not solvable and hence, a unique minimally restrictive language does not exist. In such cases, a basis is needed for language comparison to select an appropriate alternative.

We consider the case of prefix-closed regular languages containing "illegal" strings. Languages are naturally partially ordered by inclusion. The aim here is to define a measure for these languages based upon the nature of "illegal" strings they contain. The measure should impose a total ordering on such languages and thus enable selection of better controllers.

Keywords: Discrete Event Systems, Regular Languages, Finite State Automata.

1 Introduction

Following the notation in [4], $L(\mathcal{G})$ is the set of all possible finite sequence of events that can occur ; while $L_m(\mathcal{G}) \subset L(\mathcal{G})$ is a distinguished subset of these sequences that may be "marked". $L_m(\mathcal{S}/\mathcal{G})$ is the language marked by $\mathcal{S}/\mathcal{G}$ ($\mathcal{G}$ under supervision of $\mathcal{S}$), and

$$L_c(\mathcal{S}/\mathcal{G}) = L(\mathcal{S}/\mathcal{G}) \cap L_m(\mathcal{G})$$

Let $L_a, L_g \subset \Sigma^*$ be given such that

$$\phi \neq L_a \subset L_g \subset L_m(\mathcal{G})$$

L_g is interpreted as "legal behavior", L_a as "minimal acceptable behavior", i.e., control of $\mathcal{G}$ in such a way that a language smaller than L_a is generated is considered inadequate.

The Supervisory Marking Problem (SMP) is to construct a proper supervisor $\mathcal{S}$ for $\mathcal{G}$ such that

$$L_a \subset L_m(\mathcal{S}/\mathcal{G}) \subset L_g$$

The Supervisory Control Problem (SCP) is to construct a proper supervisor $\mathcal{S}$ for $\mathcal{G}$ such that

$$L_a \subset L_c(\mathcal{S}/\mathcal{G}) \subset L_g$$

When SMP or SCP is solvable, it is considered desirable that the solution be *minimally restrictive* in the sense that $L_m(\mathcal{S}/\mathcal{G})$ or $L_c(\mathcal{S}/\mathcal{G})$, considered as a sublanguage of $L_m(\mathcal{G})$, be as large as possible, subject to the constraint that it is a sublanguage of L_g [4].

When SMP or SCP are not solvable, a unique minimally restrictive language does not exist and it becomes necessary to find a criterion to select an appropriate alternative. We consider here the set of prefix-closed regular languages only. Since languages are only partially ordered by inclusion, a measure which imposes a total order is required.

2 Computation of M_i

For the purpose of the ensuing discussion, it will suffice to define a finite state automaton as the quadruple (Q, Σ, δ, q_0), where Q is the finite set of states, Σ, its finite alphabet, δ, the set of transitions and q_0, the initial state. δ is easily extensible to δ^* to be defined over strings. Since the languages we consider are all prefix-closed [4], the set of final states need not be exclusively

*Department of Computer Science, Louisiana State University, Baton Rouge, LA 70808
†Jet Propulsion Laboratory, 4800 Oak Grove Dr., MS 303-310, Pasadena, CA 91109
‡Department of Computer Science, Louisiana State University, Baton Rouge, LA 70808

0-7803-1990-7/94/$4.00 ©1994 IEEE

defined. We restrict our attention to the case where $\Sigma_i \subset \Sigma$ is the set of "illegal" symbols. Any string within the language containing a $\sigma_i \in \Sigma_i$ is rendered "illegal". Legal as well as illegal strings (illegal with respect to L_g) are in the language. [3] contains a good exposition on regular languages.

Comparing languages depending upon the proportion of illegal strings they have is likely to prove worthless because languages could have an infinite number of illegal strings. We consider their *unique* minimal finite state representations instead. Then, we calculate the probability of the machine generating illegal strings. We have the following definitions :

Definition 1. A string is called *loop-free* if the corresponding unique minimal finite state graph (machine) that generates it does not visit the same vertex (state) twice to generate it.

Definition 2. I denotes the set of loop-free illegal strings. For an n-state machine,

$$I : \{sx \mid s \in \{\Sigma - \Sigma_i\}^* \text{ and } loop-free,\ x \in \Sigma_i, |sx| \leq n-1\}$$

Any string, having a member of I as its prefix, is illegal too. It follows from the definition that any proper prefix of a string $s \in I$ is legal. We now define a *measure of illegality*, M_i, of a language L, such that,

$$M_i(L) = Pr(I) \tag{1}$$

where $Pr(I)$ is the probability that the machine generates strings in I. Let $s = s_1 s_2 .. s_l$ be a string in I. Then,

$$Pr(I) = \sum_{s \in I} Pr(s) \tag{2}$$

$$Pr(s) = Pr(s_1)Pr(s_2)..Pr(s_l), \text{and,} \tag{3}$$

$$Pr(s_i) = \frac{Pr(q_i)}{|\delta(q_i, *)|} \tag{4}$$

where, $Pr(q_i)$ is the probability that the machine is in state q_i before the generation of s_i, $\delta(q_i, s_i)$ is defined, and $|\delta(q_i, *)|$ denotes the number of transitions defined from the state q_i.

Definition 3. A state q_i is primary-illegal with respect to a string $s \in I$, if $\delta^*(q_0, s) = q_i$.

If we consider strings of length 0, the probability that the machine is in the initial state is 1. q_0 denotes the initial state. Let Q_i denote the set of states reachable from q_0 by paths of length i, i.e.,

$$Q_i = \{q | \delta(q_0, s) = q, s \in I, |s| = i\}.$$

From the above definitions, we have the following proposition.

Proposition 1. The sum of the probabilities of strings of any given length is 1. For all k,

$$\sum_{s \in L, |s|=k} Pr(s) = 1. \tag{5}$$

Proof. We prove the proposition by induction.

$Basis(n = 0)$. For the empty string, ϵ, $Pr(q_0) = 1$.

$(n = 1)$. For each s_i, such that $\delta(q_0, s_i)$ is defined,

$$Pr(s_i) = \frac{Pr(q_0)}{|\delta(q_0, *)|} \tag{6}$$

$$Pr(s_i) = \frac{1}{|\delta(q_0, *)|}. \tag{7}$$

Therefore, $\Sigma_{s \in L, |s|=1} Pr(s) = 1$. It also follows from the above that,

$$\sum_{q \in Q_1} Pr(q) = \sum_{s \in L, |s|=1} Pr(s) = 1. \tag{8}$$

$Hypothesis(n = k)$. Assume $\sum_{|s|=k} Pr(s) = 1$.

$InductionStep(n = k + 1)$. Let the string, sx_i, be of length k+1. After the execution of s, the machine is in $q \in Q_k$. Then, we have

$$\forall q, q \in Q_k, \sum_{x_i, \delta(q, x_i) is defined} Pr(x_i) = Pr(q) \tag{9}$$

But it is known that, $\sum_{q \in Q_k} Pr(q) = 1$. Hence, it follows that,

$$\sum_{s \in L, |s|=k+1} Pr(s) = \sum_{q in Q_{k+1}} Pr(q) = 1. \qquad (10)$$

Thus proved.■

It is evident that the above measure gives a quantitative comparison basis between any pair of languages and if $M_i(L_1) > M_i(L_2)$, L_2 has a lesser probability of generating "illegal" strings than L_1 and hence is preferable.

3 An algorithm to compute M_i

For the algorithm *compute_*M_i, we will consider a directed graph $G = (V, E)$ to represent the finite state automaton, where V is the set of vertices (states) and E is the set of directed edges (transitions). *compute_*M_i is actually a modified version of the traditional depth first search (dfs) algorithm [1]. Its first visit to the vertices is in the same order as dfs , but it differs from dfs in the following respects :

1. After traversing an edge with label $\sigma_i \in \Sigma_i$, it backtracks.

2. There is no restriction on the number of times an edge may be traversed.

3. It does not traverse edges that complete cycles in the graph.

The basic idea is to traverse edges till either a loop is completed or an element of Σ_i is encountered. The search is exhaustive in that every element of I is traversed. Backtracking is done the same way as in dfs. For graph theoretic concepts, the reader is referred to [2]. A proof of correctness is also given.

mark_state(q_i) : an array of flags to indicate if state q_i is primary-illegal. A value of 1 means it is primary-illegal and 0 means it is not.

edge_prob$(1..|E|)$: an array of numbers for the set of edges of the graph denoting the probability that the edge is traversed.

traversed$(1..|E|)$: an array of flags for the set of edges to indicate if a particular edge has been traversed or not. 0 indicates that it has not been traversed, 1 indicates it has.

state_prob$(1..|V|)$: an array of numbers for the set of vertices of the graph denoting the probability that the state is visited.

curr_state : the current vertex being visited.

$O(q_i)$: the set of outgoing edges at vertex q_i.

$I(q_i)$: the set of incoming edges at vertex q_i.

1. For all vertices $v \in V$, *mark_state(v)* = *state_prob(v)* = 0.

2. For all edges $e \in E$, initialize *edge_prob*$(e_i) = 0$.

3. *curr_state* $= q_0$.

4. For all $e \in O(curr\_state)$, *traversed*$(e) = 0$. For all $q \in \delta(curr\_state, e)$,

   mark_state$(\delta(curr\_state, e)) = 0$.

5. Select an outgoing edge e and set *traversed*$(e) = 1$. If all the edges incident upon *curr_state* have been traversed, backtrack. If no vertices are left, stop.

6. Set *traversed*$(e) = 1$. If e is a results in a cycle, go to step 4. If $e \in \Sigma_i$, then *edge_prob*$(e) = \frac{1}{|\delta(curr\_state,*)|}$, *mark_state*$(\delta(curr\_state, e)) = 1$. Otherwise, *edge_prob*$(e) = \frac{1}{|\delta(curr\_state,*)|}$; *curr_state* $= \delta(curr\_state, e)$, and go to step 3.

7. After the graph has been traversed, the edge probabilities along a path are to be multiplied. The sum of the path probabilities of the paths ending in primary-illegal vertices evaluate to M_i, i.e.,

$$M_i = \sum_{s \in I} Pr(s).$$

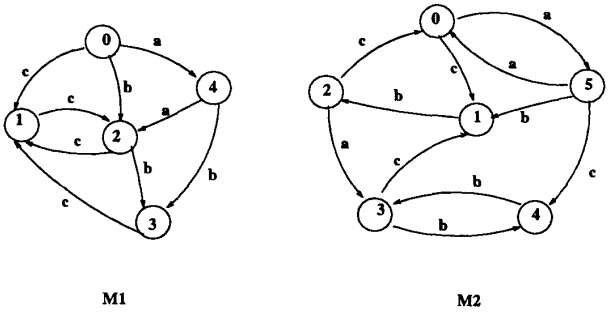

Figure 1: M1 and M2.

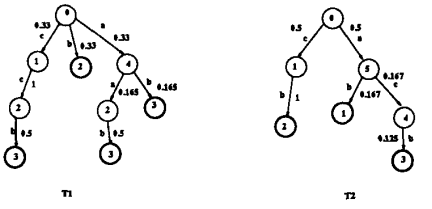

Figure 2: T1 and T2.

Proof of Correctness. Step 1 to 4 are initialization steps. All the outgoing edges incident upon a vertex are selected one after the other to be traversed in Step 5. Their respective subtrees are traversed (depth first search) first. Step 6 is the main step it checks if the edge results in cycle, it is not updated and it goes on to the next edge; if the edge is an element of Σ_i, then the adjacent vertex is marked primary-illegal. Since any vertex is primary- illegal only with respect to a particular string, Step 4 reinitializes the *mark_state* for all adjacent vertices. The algorithm terminates because it can detect loops, has finite states and because of Step 5.■

4 An example

Let the consider the two finite state graphs in Figure 1. For both the graphs, $\Sigma = \{a, b, c\}$ and $\Sigma_i = \{b\}$. Figure 2 shows the details of the computation.

In these examples, all the leaves of T1 and T2 are primary-illegal. This need not be the case always. From the values shown in the figures, $M_i(L1) = 0.33 \times 1 \times 0.5 + 0.33 + 0.33 \times 0.165 \times 0.5 + 0.33 \times 0.165 = 0.768$, and $M_i(L2) = 0.5 \times 1 + 0.5 \times 0.167 + 0.5 \times 0.167 \times 0.125 = 0.594$. From these considerations, L2 is preferable to L1.

5 Conclusions

We introduced a measure to allow comparison among controllers when a unique "best" one does not exist. We restricted our attention to the case where the illegal symbols are a subset of the alphabet. A possible generalization would be to describe the "illegal" strings themselves as a regular language, and not restrict them to being a subset of the alphabet. The extension to non-regular languages may be more difficult. For non-regular languages, the computability of M_i can be examined under some restrictions, such as the finiteness of the primary-legal states, the finiteness of the outgoing edges at all states, etc.

Acknowledgement

The authors wish to thank the anonymous refree for his comments and suggestions.

References

[1] A. Aho, J.E. Hopcroft and J.D. Ullmann, *"Design and Analysis of Algorithms"*, Addison-Wesley, Reading, Mass., 1974.

[2] N. Deo, *"Graph Theory with applications to Engineering and Computer Science"*, Prentice Hall Inc., New Jersey, 1974.

[3] J.E. Hopcroft and J.D. Ullman, *"Formal Languages and their Relation to Automata*, Addison-Wesleay, Reading, Mass., 1969.

[4] P.J. Ramadge and W.M. Wonham, "Supervisory control of a class of discrete event processes", *SIAM J. Control and Optimization*, **25**(1), pp. 206 230, 1987.