Reachability and Recurrence in Extended Finite State Machines: Modular Vector Addition Systems

A.S. Krishnakumar

AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey 07974, USA

Abstract. In this paper, a formal model of Extended Finite State Machines (EFSMs) is proposed and an approach to their analysis is suggested. The state of an EFSM is captured by its *configuration*. A class of EFSMs, called Modular Vector Addition Systems (MVAS), is defined and analyzed. Modular Vector Addition Systems cover a significant subset of models used in communication protocols and behavioral synthesis of hardware. For this class of EFSMs, an algorithm to compute the set of configurations reachable from an initial configuration is presented. This algorithm may also be used to compute the set of *recurrent* configurations. Knowledge of these sets is useful in verification, testing, and optimization of EFSM models. A compact representation of these sets and a simple test for membership for such representations are also presented.

1 Introduction

This paper introduces a formal model for Extended Finite State Machines (EFSMs). The EFSM model is used widely in behavioral specification of hardware models and in communication protocol specifications. It is notationally compact and has been used as the basis of many specification languages, e.g. ESTELLE [1] and APSL [2]. However, most attempts to analyze these models have been based on conventional FSM techniques. This approach suffers from the state explosion problem and does not exploit any algebraic relations that may be present in the specification. A procedure for analyzing EFSMs without going to a conventional FSM representation is suggested in this paper. One step in this procedure is adapted from a technique (called *stabilization*) developed for transition systems that was reported in [3, 4, 5]. Using the EFSM model, an algorithm to generate functional test vectors automatically for a class of hardware models is described in [6]. These results indicate that there is an advantage in treating EFSMs as a mathematical object instead of simply a notational convenience. In the next section the Extended Finite State Machine model will be introduced and the terms used in its analysis will be defined. Section 3 introduces Modular Vector Addition Systems and develops their analysis. Section 4 provides an example of the results of Section 3. Concluding remarks are made in Section 5.

2 Extended Finite State Machines

2.1 The Model

A Extended Finite State Machine M is defined as a 5-tuple $\{S, I, O, D, T\}$,where

S = a finite set of nodes,
I = a set of input symbols,
O = a set of output symbols,
D = an n-dimensional linear space,
T = a transition relation, $T : S \times 2^D \times I \rightarrow 2^{(S \times D)} \times O$.

Points in D will be denoted by n-tuples $\mathbf{x} = (x_1, \ldots, x_n)$. We call $x_i, i = 1, \ldots, n$, the *variables* associated with M. It is not necessary that D be finite or even countably infinite(*e.g.* consider $D = \Re^n$). Therefore, a general EFSM need not have an equivalent finite state model. The ordered pair $< s, \mathbf{x} > \in S \times D$ will be called a *configuration* of M. The operation of M is to be understood as follows: If the machine is in configuration $< s, \mathbf{x} >$, on receiving an input i, it moves to a configuration $< t, \mathbf{y} >$ if and only if there exist P, $Q \in 2^D$ with $\mathbf{x} \in P$ and $\mathbf{y} \in Q$ such that $((s, P, i), (t, Q, o)) \in T$ for some $o \in O$. In this case, we say that there is a *transition from s to t labeled with the enabling function(or predicate) f and update function u*, where $f(\mathbf{x}) = 1$, if $\mathbf{x} \in P$ and 0 otherwise, and $u(.)$ is the function that maps P to Q. Note that the *domain* of u is P. We denote this transition by $s \xrightarrow{(f,u)} t$. The notation $u_i(x)$ will refer to the i^{th} component of $u(\mathbf{x})$. An update function u is *applicable* at a configuration $< s, \mathbf{x} >$, if there is a transition $s \xrightarrow{(f,u)} t$ such that $f(\mathbf{x}) = 1$. The *transition graph* G of M has one node for each $s \in S$ (labeled s) and an edge between s and t if and only if there exist P, $Q \in 2^D$, $i \in I$, and $o \in O$ such that $((s, P, i), (t, Q, o)) \in T$. We associate f, u, i, and o with this edge.

A sequence of configurations $< s_i, \mathbf{x}_i >, i = 1, \ldots, n$, is *admissible* if, for $j = 1, \ldots, n-1$, there exist $i_j \in I$ and $o_j \in O$ such that $((s_j, \mathbf{x}_j, i_j), (s_{j+1}, \mathbf{x}_{j+1}, o_j)) \in T$; otherwise, the sequence is *inadmissible*. If a sequence $< s_i, \mathbf{x}_i >, i = 1, \ldots, n$ is admissible, then $s_1 s_2 \ldots s_n$, considered as vertices of G, form a path in G. However, the converse is not true, *i.e.*, given a configuration $< p_0, \mathbf{w}_0 >$ and a path $p_0 p_1 \ldots p_k$ in G, let us construct $< p_i, \mathbf{w}_i >$, for $i = 1, \ldots, k$ in an obvious manner. Then, the sequence of configurations $< p_i, \mathbf{w}_i >$ for $i = 0, \ldots, k$ is not necessarily admissible. This is due to the fact that even though for some $\alpha_i \in I$ and $\beta_i \in O$ $((p_i, w_j, \alpha_i), (p_{i+1}, w_{j+1}, \beta_i)) \in T$ (by virtue of p_i and p_{i+1} being successive nodes of a path in G), and there is an edge from p_{j+1} to p_{j+2} in G, it does not necessarily follow that $((p_{i+1}, w_{j+1}, \alpha_{i+1}), (p_{i+2}, w_{j+2}, \beta_{i+1})) \in T$. A configuration $< q, \mathbf{y}_f >$ is *reachable* from another configuration $< p, \mathbf{y}_0 >$ if there exists a sequence $< s_i, \mathbf{y}_i >, i = 1, \ldots, k$ such that $< p, \mathbf{y}_0 >< s_1, \mathbf{y}_1 > \ldots < s_k, \mathbf{y}_k >< q, \mathbf{y}_f >$ is admissible. If $< q, \mathbf{y}_f >$ is reachable from $< p, \mathbf{y}_0 >$ and *vice versa*, they are *biconnected*. A configuration $< p, \mathbf{y} >$ is *recurrent* if $< p, \mathbf{y} >$ and the initial configuration $< s_0, \mathbf{x}_0 >$ are biconnected.

We can separate the set of variables X as $X_c \cup X_{\overline{c}}$ where X_c is the set of variables that appear in at least one enabling function on some transition and $X_c \cap X_{\overline{c}} = \phi$ ($\mathbf{x} = [\mathbf{x}_c\ \mathbf{x}_{\overline{c}}]'$). This induces an obvious decomposition of D into $D_c \times D_{\overline{c}}$, where $\mathbf{x}_c \in D_c$ and $\mathbf{x}_{\overline{c}} \in D_{\overline{c}}$. We call variables in X_c as *control variables* and variables in $X_{\overline{c}}$ as *non-control variables.*

2.2 Analysis of EFSMs

Given an EFSM M, we may apply a technique called *stabilization* to derive another EFSM M' such that *every path in the transition graph of M' is an admissible path.* This property greatly simplifies the analysis of EFSMs as it allows us to study admissible sequences by studying the paths of $G(M')$ (It is always true that the nodes in an admissible sequence form a path in the transition graph. The converse is not true for general EFSMs). A *stabilized* EFSM will be called a Vector Transformation System (VTS). The stabilization technique that we use for EFSMs is adapted from the one described in [5], which also describes the properties of this algorithm. Note that stabilization concerns only the *control* variables of the EFSM — the *non-control* variables do not play a part. However, in some applications, it is of interest to consider the reachability of non-control variables also. In general, the stabilization algorithm need not terminate.

In the sequel, we will study a special case where the update operations of the stabilized system are limited to additions, subtractions, or constant assignments and the enabling predicates are limited to comparisons of single variables to constant threshold values. In this case, the stabilization algorithm terminates resulting in a Vector Addition System.

3 Modular Vector Addition Systems

A Vector Addition System is a special case of a Vector Transformation System (VTS) whose transitions have update functions u_{ij} such that:

$$u_{ij}(\mathbf{x}) = \mathbf{x} + \mathbf{b}_{ij} \tag{1}$$

where $\mathbf{b}_{ij} \in D$ is a constant vector. We call $\mathbf{b}_{ij}$ the *edge vector* of the transition. A Modular Vector Addition System (MVAS) is a Vector Addition System in which $D = \mathbf{Z}_{m_1} \times \cdots \times \mathbf{Z}_{m_n}$ and all arithmetic for the i^{th} component of $\mathbf{x}$ is performed *modulo* m_i. However, we can derive an equivalent MVAS for which $D = \mathbf{Z}_m \times \cdots \times \mathbf{Z}_m$. This follows from the fact that $x \equiv b\ mod\ m$ and $kx \equiv kb\ mod\ km$ have the same solution set for x. Thus, taking m to the *least common multiple* of all the moduli, we can transform the system to an equivalent one (in that they have the same solution set) in which all equations have the same modulus. Therefore, the MVAS in the sequel will be assumed to have a single modulus m[1].

[1] If the m_i are arbitrary, their least common multiple (lcm) can be quite large. However, in many applications of interest the moduli are powers of 2 and hence their lcm is simply the largest modulus.

The study of MVAS is motivated by hardware models. Many hardware models make use of unsigned integers which are realized as registers of finite width. Thus all arithmetic on these registers is naturally modular and the common modulus is simply $\max(m_i)$. In the following sections, we develop some properties of MVAS and then derive an explicit characterization of the set of recurrent configurations for a specified initial configuration $< s_0, \mathbf{x}_0 >$. Previous work on reachability in VAS imposed the condition that the path connecting the configurations lie entirely in the positive orthant of n–space (*see e.g.* [7]). This restriction is not relevant here and thus we have a "simpler" problem.

3.1 Analysis Using Simple Cycles

In this section we show that reachability and recurrence of *configurations* are equivalent concepts within a strongly connected component of the transition graph G of a MVAS V. Thus, reachability analysis can be done using the cycles of a strongly connected component of the graph G. For any cycle, the sum of its edge vectors will be called a *cycle vector.* A *simple cycle* of G is any path $q_0 q_i \ldots q_n q_0$ such that, for $i = 1, \ldots, n$, $q_i \neq q_0$ and $q_i \neq q_j$ for $i \neq j$. The cycle vector of a simple cycle is a *simple cycle vector.*

Lemma 1. *Let p and q be two biconnected nodes in G. If $< q, \mathbf{y} >$ is reachable from $< p, \mathbf{x} >$, then $< p, \mathbf{x} >$ is reachable from $< q, \mathbf{y} >$.*

Proof: Adjoin the path from q to p to the path taking $< p, \mathbf{x} >$ to $< q, \mathbf{y} >$. This is a cycle in G with a cycle vector Δ *i.e.* $\mathbf{x} \rightarrow \mathbf{x} + \Delta$. By traversing this cycle a *finite* number of times ($\leq m$), we can make the total added vector to be $\mathbf{0}$. This brings us back to the configuration $< p, \mathbf{x} >$. Hence the result. ■

Lemma 2. *Let p and q belong to the same strongly-connected component of G. Then, if $< p, \mathbf{y} + \Delta >$ is reachable from $< p, \mathbf{y} >$, then $< q, \Delta >$ is reachable from $< q, \mathbf{0} >$.*

Proof: Since $< p, x + \Delta >$ is reachable from $< p, y >$, there is a cycle C_1 beginning and ending at p whose cycle vector is Δ. Further, p and q lie on a cycle C_2 with cycle vector Δ_1 since they are in the same strongly-connected component of G. Consider the following path: q to p along C_2, C_1, p to q along C_2. The cycle vector of this cycle is $\Delta + \Delta_1$. Now traverse C_2 k ($\leq m$) times such that $(k+1)\Delta_1 = \mathbf{0}$. The cycle vector of this cycle beginning and ending at q is clearly Δ. Hence the result. ■

In the following we assume G is strongly connected . Let us consider the set of configurations at node p reachable from an initial configuration $< p, \mathbf{x}_0 >$. This is obtained by computing the final vector $\mathbf{x}_f$ for every possible cycle of G rooted at p. Lemma 1 shows that this is also the set of recurrent configurations. Suppose we start at the configuration $< p, \mathbf{x}_0 >$ and traverse a simple cycle rooted at p with a final configuration $< p, \mathbf{x}_f >$. In an MVAS, the vector $(\mathbf{x}_f - \mathbf{x}_0)$ is independent of $\mathbf{x}_0$ and depends only on the simple cycle (in fact, it is the cycle vector). There are only a *finite* number of simple cycles(not necessarily containing p) in G and

any cycle of G can be expressed as a composition of these cycles. Therefore, the cycle vector of any cycle can be expressed as a *linear combination* of the simple cycle vectors. Thus, given $< p, \mathbf{x}_f >$ reachable from $< p, \mathbf{x}_0 >$, we can write

$$(\mathbf{x}_f - \mathbf{x}_0) \equiv \sum_{\text{C a simple cycle}} r_c \mathbf{a}_c \ mod\ m \tag{2}$$

where $\mathbf{a}_c$ is the cycle vector for simple cycle C and r_c is the number of times C occurs in the decomposition of the cycle into simple cycles. If (2) has a solution, then Lemma 2 shows that we can construct a solution in which the coefficient of any specific $\mathbf{a}_c$ is non-zero. Thus, if the difference $(\mathbf{x}_f - \mathbf{x}_0)$ can be generated by some simple cycles in G, it can always be generated by a cycle passing through a specified node of G. Therefore, every configuration $< p, \mathbf{y} >$ reachable from $< p, \mathbf{x}_0 >$ is such that $\mathbf{y} \equiv (\mathbf{x}_0 + \sum r_c \mathbf{a}_c)\ mod\ m$, where the sum runs over the simple cycles of G. This can be written as $\mathbf{y} \equiv (\mathbf{x}_0 + A\mathbf{v})\ mod\ m$, where A is a *cycle vector matrix of* G whose columns are the simple cycle vectors of G and $\mathbf{v}$ is a vector whose i^{th} element is v_i. Therefore, the set of reachable configurations $< p, \mathbf{y} >$ is given by $\{\mathbf{y} \mid \mathbf{y} \equiv (\mathbf{y}_0 + A\mathbf{v}) mod\ m$ for arbitrary $\mathbf{v}\}$and is denoted by $R(\mathbf{y}_0, A)$. [2]

If $R(\mathbf{y}_0, A)$ is a set of reachable configurations at p, it gives rise to a set of reachable configurations $R(\mathbf{y}_{0,q}, A)$ at any other node q. Here $\mathbf{y}_{0,q}$ is the final vector obtained by following any path from p to q with initial value $\mathbf{y}_0$ at p. We can derive these sets systematically by finding a cycle in G that passes through all the nodes of G at least once. Then, given a set $R(\mathbf{y}_0, A)$ at some node in G, $R(\mathbf{y}_{0,q}, A)$ for other nodes q in G can be found by following this cycle. We have developed a simple test to decide if a given $\mathbf{y}$ belongs to $R(\mathbf{y}_0, A)$. In order to develop this test, we need some results from the theory of Linear Congruences. A good introductory reference for this material is [8]. The details of this test are given in the Appendix.

If G has more than one strongly-connected component, this method can be extended by computing a cycle vector matrix for each component. Initial point(s) for components other than the one containing the specified initial configuration can be computed after ordering them using a topological sort.

3.2 MVAS With Constant Assignments

Quite often one encounters transitions in an MVAS which assign a constant value to a variable. These are commonly found in hardware models — in fact, it is not possible to initialize hardware systems without such operations. We call such systems *MVAS with constant assignments* (MVAS-CA). The update function u_{ij} of a transition t_{ij} of an MVAS-CA can be represented as $u_{ij} \equiv (D_{ij}\mathbf{x} + \mathbf{b}_{ij})\ mod\ m$, where D_{ij} is a diagonal matrix whose diagonal elements

[2] The number of cycles could be very large. But the cycle vector matrix can be efficiently computed by keeping only *linearly independent* vectors as each cycle is found. If the rank of the matrix becomes full, no further computation is needed as all the values are reachable.

$\in \{0,1\}$. If every D_{ij} is the identity matrix, then we have a simple MVAS. In MVAS-CA, if a cycle transformation contains a constant assignment to a variable, then independent of the number of times the cycle is traversed (as long as it is greater than zero), the same constant value is assigned to that variable.

In this paper, we consider only those MVAS-CA in which each D_{ij} is either the *identity* or the *zero* matrix. In other words, either all variables are assigned constant values or none. We call such systems *regular* MVAS-CA. The transitions of a regular MVAS-CA are of two types - *regular* and *constant-assigning.* A transition from s_i to s_j for which the corresponding D_{ij} is the identity is called a *regular transition.* Otherwise, it is a *constant-assigning transition* (CA-transition).

3.3 Analysis of Regular MVAS-CA

In this section, we will describe a procedure to compute reachable and recurrent configurations of a regular MVAS-CA. Recall that $R(\mathbf{x}, A) = \{\ \mathbf{y} \mid \mathbf{y} \in D$ and $\mathbf{y} \equiv (\mathbf{x} + Av)\ mod\ m\}$. The notation $[A_1 \mid A_2]$ will denote the matrix formed by adjoining the columns of A_2 to those of A_1.

Analysis Considering only the *regular* transitions of G, we find its strongly connected components (if there are no CA-transitions this will, of course, be all of G). Let these be called $G_1, G_2, \ldots, G_k$. By convention, G_1 will be contain the initial node. For each G_i, we calculate a matrix A_i whose columns are the cycle vectors of G_i. If G_i is an isolated node, we take A_i to be the scalar 0. With each node q of G we associate a set S_q of ordered pairs $(\mathbf{x}, A)$, where $\mathbf{x} \in D$ and A is a matrix. The set of reachable configurations at each node q will then be given by $\bigcup_{S_q} R(\mathbf{x}, A)$. Our analysis provides a procedure for computing S_q for every q in G.

We construct a new graph H which is obtained from G by replacing each G_i with a single node v_{G_i}. The remaining nodes are left unchanged. A transition incident or leaving a node in G_i in G will be incident or leave from v_{G_i} in H. Now, H can have *no cycle* consisting only of *regular* transitions that passes through some v_{G_i}. For, if there is such a cycle, then all the G_i corresponding to the v_{G_i} in this cycle along with the transitions forming the cycle would form a strongly connected component of G consisting only of *regular* transitions. This contradicts the assumption that each G_i is obtained as a strongly connected component of G considering all the *regular* transitions of G. Therefore, H restricted to its regular transitions must be a directed acyclic graph (dag). Based on this dag, we can rank the vertices of H such that for any two nodes p and q of H, $rank(p) < rank(q)$ iff there is a path from p to q consisting only of regular transitions. We assign ranks to nodes in G_i by assigning the same rank $rank(v_{G_i})$ to every node in G_i. There are two kinds of paths in H — regular and CA. A path that contains at least one CA-transition is a *CA path.* The path vector of such a path is a CA-vector. If a path has only regular transitions, it is a *regular path* and its *path vector* is the sum of the edge vectors of the transitions in the path.

Given a regular MVAS-CA V with initial configuration $< s_0, \mathbf{x}_0 >$, we construct the graph H as described above and rank the vertices of $G(V)$ based on H. We then proceed to compute S_q for every node q in G. An algorithm to do this is given in Figure 1. The first phase accounts for the CA-transitions in H and the second the regular transitions.

For all nodes q, set $S_q = \phi$.
Set $S_{v_{G_1}} = \{(\mathbf{x}_0, A_1)\}$.

/* Phase I : Deal with CA-transitions */

for (*each node* v_{G_i} *in* H) **do begin**
  for (*each node* v_{G_j} *such that* $rank(v_{G_j}) < rank(v_{G_i})$) **do begin**
    if(*there is a CA-transition from* v_{G_j} *to* v_{G_i})**then**
      Add the image set under this CA-transition to the set of reachable configurations at p, where p is the head node (in G) of the CA-transition.
    endif
  end
end
Remove all CA-transitions from H.

/* Phase II : Deal with regular transitions */

for (*each node* v_{G_i} *such that* $rank(v_{G_i}) < rank(v_{G_1})$) **do begin**
  for (*each node* v_{G_j} *such that* $rank(v_{G_j}) = rank(v_{G_i} - 1)$) **do begin**
    if(*there is a transition from* v_{G_j} *to* v_{G_i})**then**
      For each set of reachable configurations at p, add its image under this regular transition to the set of reachable configurations at q. (p is the tail node and q the head node (in G) of this transition)
    endif
  end
end

Fig. 1. Algorithm for Computing Reachable Configurations

In Phase I, we update the set of configurations at every node by finding the images under every CA-transition. This is done according to the discussion below on computing image sets. If the CA-transition is from v_{G_j} to v_{G_i} (p to q in G), we use the image at q to update the set of configurations at other nodes in G_i as described in Section 3.1It can be shown that every one of the starting points computed in Phase I is reachable from the initial configuration. Before continuing with Phase II, we remove all CA-transitions from the graph.

Phase II finds the image of the sets of ordered pairs $(\mathbf{x}, A)$ describing reachable configurations at the nodes v_{G_i} found in Phase I under regular transitions

according to the discussion in Section 3.3. When the two phases are complete, we have a set of ordered pairs $(\mathbf{x}, A)$ for each node in G that defines its set of reachable configurations. More importantly, it can be shown that any two configurations $< p, \mathbf{x} >$ and $< q, \mathbf{y} >$ from these sets (except perhaps $R(\mathbf{x}_0, A_1)$) are strongly connected. In order for a reachable configuration to be recurrent, there must be a path from it to the initial configuration $< s_0, \mathbf{x}_0 >$. Therefore, we check if any of the sets at the initial node intersects the set of recurrent configurations $R(\mathbf{x}_0, A_1)$. If there is an intersection, the set of reachable configurations computed above is also the set of recurrent configurations. The results of Lemmas 3 and 4 may be used to simplify this set. If there is *no* intersection, then the set of recurrent configurations is simply $R(\mathbf{x}_0, A_1)$.

Given a configuration $< q, \mathbf{x}_q >$, we test if it is in the set of reachable configurations as follows:

for(*each* $(\mathbf{x}, A)$ *in* S_q **) do begin**
  if ($\mathbf{x}_q \in R(\mathbf{x}, A)$ **) then**
    report *true* and exit.
  endif
end
report *false* and exit.

In a similar manner, we can test for recurrent configurations. This completes the description of the sets of reachable and recurrent configurations of regular MVAS-CA.

Image of a Set of Configurations Consider two nodes p and q such that there is a transition e from p to q. Let $R(\mathbf{x}_1, A_p)$ be a set of configurations at p. Suppose e is a regular transition from p to q with edge vector $\mathbf{x}_d$. Then the *image* of $R(\mathbf{x}_1, A_p)$ under e is the set $\{< q, \mathbf{x} > |\ \mathbf{x} = \mathbf{u} + \mathbf{x}_d + A_q\mathbf{v},\ \mathbf{u} \in R(\mathbf{x_1}, A_p)\}$. This can be written more compactly as $\{< q, \mathbf{x} > |\ \mathbf{x} \in R(\mathbf{x_1} + \mathbf{x}_d, [A_p \mid A_q])\}$. If e is a CA-transition assigning the constant vector $\mathbf{k}$, then the image is the set $\{< q, \mathbf{x} > |\ \mathbf{x} = \mathbf{k} + A_q\mathbf{v}\}$. Note that this image is *independent* of the set $R(\mathbf{x}_1, A_p)$ at p. We now state two lemmas that can be easily verified by algebraic manipulation. All the matrices and vectors in this discussion will be assumed to have dimensions appropriate to the multiplications and additions indicated. All arithmetic is to be understood as that of $\mathbf{Z}_m$.

Lemma 3. *Given* $\mathbf{x}_1, \mathbf{x}_2 \in D$ *and a matrix* A, one and only one *of the following statements is true:*

1. $R(\mathbf{x}_1, A) = R(\mathbf{x_2}, A)$
2. $R(\mathbf{x}_1, A) \cap R(\mathbf{x_2}, A) = \phi$.

Lemma 4. *Given* $\mathbf{x}_1, \mathbf{x}_2$, $\mathbf{x}_d$, *all in* D, *and matrices* A_1 *and* A_2, one and only one *of the following statements is true:*

1. $R(\mathbf{x}_1 + \mathbf{x}_d, [A_1 \mid A_2]) \supseteq R(\mathbf{x}_2, A_2)$
2. $R(\mathbf{x}_1 + \mathbf{x}_d, [A_1 \mid A_2]) \cap R(\mathbf{x}_2, A_2) = \phi$.

4 An Example of an MVAS-CA

In this section, we will consider an example of an MVAS-CA. This system V is shown in Figure 2. The set S of nodes of V is $\{S_1, S_2, S_3, S_4, S_5, S_6\}$. $G(V)$

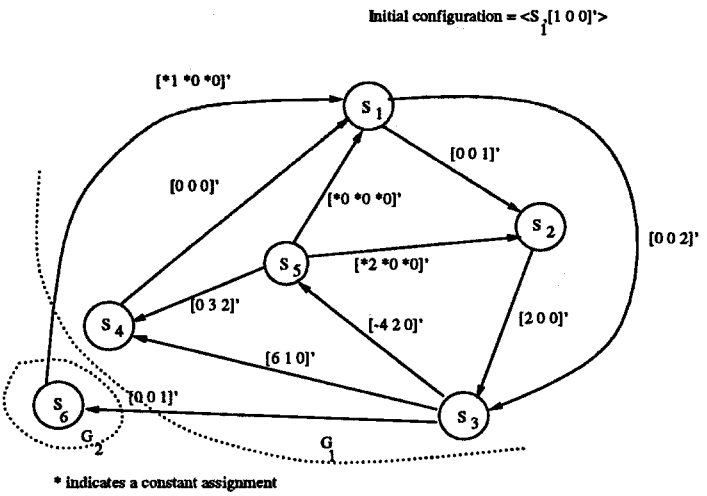

Fig. 2. An MVAS with Constant Assignments

is the directed graph in Figure 2. A prime is used to indicate a transpose — $[000]'$ indicates a column vector. An edge labeled $[001]'$ from S_1 to S_2 indicates that $u(\mathbf{x}) = \mathbf{x} + [001]'$. In other words, it is the edge vector. Labels whose entries are marked with an asterisk are CA-transitions. Thus, the edge from S_5 to S_1 marked $[*0 * 0 * 0]'$ means that $u(\mathbf{x}) = [000]'$ for this transition. Since all the transitions assign constant values to either all variables or none, this is a regular MVAS-CA. A common modulus $m = 16$ is assumed for this system. The initial configuration is taken as $< S_1, [100]' >$. The strongly connected components of G, considering only the regular transitions, are enclosed in dotted lines in Figure 2. The component containing the initial node S_1 is G_1. G_2 is an isolated node S_6. The derived graph H is shown in Figure 3. The nodes of H are ranked as $rank(v_{G_1}) = 0$ and $rank(v_{g_2}) = 1$. There are four simple cycles in this component - $S_1S_2S_3S_4S_1$, $S_1S_3S_4S_1$, $S_1S_2S_3S_5S_4S_1$, and $S_1S_3S_5S_4S_1$. All cycles in this component can be expressed as the composition of these simple cycles. The cycle vectors of this component are $[811]'$, $[612]'$, $[-253]'$, and $[-454]'$. Therefore, the cycle vector matrix A_1 of G_1 is

$$A_1 = \begin{pmatrix} 8 & 6 & -2 & -4 \\ 1 & 1 & 5 & 5 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

and the cycle vector matrix A_2 of G_2 is $\mathbf{0}$. At the beginning of Phase I, we have the following sets of starting points and matrices:

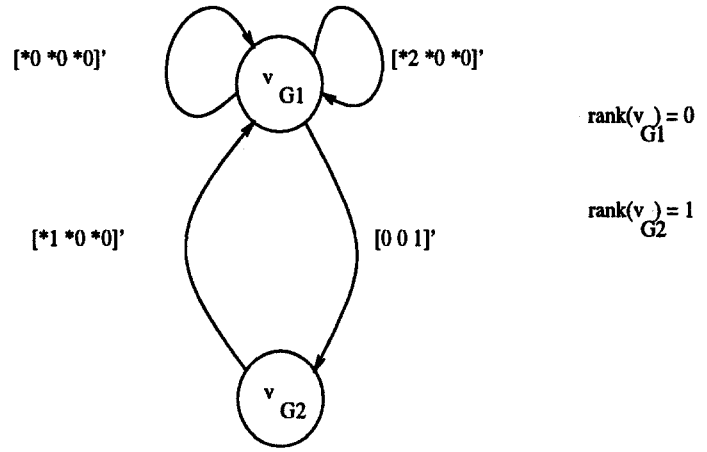

Fig. 3. Derived graph H for the example MVAS-CA

$$S_{S_1} : \{([100]', A_1)\} \quad S_{S_2} : \{([101]', A_1)\}$$
$$S_{S_3} : \{([301]', A_1)\} \quad S_{S_4} : \{([-153]', A_1)\}$$
$$S_{S_5} : \{([-121]', A_1)\} \; S_{S_6} : \{(\phi, 0)\}$$

The starting points for G_1 were obtained by following the cycle $S_1S_2S_3S_5S_4S_1$. We now consider the CA-transitions of G. The transition from S_5 to S_2 with CA-vector $[*2*0*0]'$ adds $([200]', A_1)$ to S_{S_2}. This leads to the following starting points at the other nodes in G_1: $[400]'$ at S_3, $[020]'$ at S_5, $[052]'$ at S_4, and $[052]'$ at S_1. After Phase I is completed, we have the following:

S_{S_1} : $\{([100]', A_1), ([052]', A_1), ([000]', A_1), ([100]', A_1)\}$;
S_{S_2} : $\{([101]', A_1), ([200]', A_1), ([001]', A_1), ([101]', A_1)\}$;
S_{S_3} : $\{([301]', A_1), ([400]', A_1), ([201]', A_1), ([301]', A_1)\}$;
S_{S_4} : $\{([-153]', A_1), ([052]', A_1), ([-253]', A_1), ([-153]', A_1)\}$;
S_{S_5} : $\{([-121]', A_1), ([020]', A_1), ([-221]', A_1), ([-121]', A_1)\}$;
S_{S_6} : ϕ;

We have one regular transition to account for in Phase II — from S_3 to S_6. Performing this update we get

S_{S_6} : $\{([302]', A_1), ([401]', A_1), ([202]', A_1), ([302]', A_1)\}$.

We now have the sets of reachable configurations at each node. We check if any of the sets at the initial node S_1 overlaps $R([100]', A_1)$. First, we transform A_1 to an upper-triangular matrix A^* by pre-multiplying it by a matrix T:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & -1 & 1 \\ 1 & -10 & 2 \end{pmatrix} \begin{pmatrix} 8 & 6 & -2 & -4 \\ 1 & 1 & 5 & 5 \\ 1 & 2 & 3 & 4 \end{pmatrix} = \begin{pmatrix} \boxed{1} & 1 & 5 & 5 \\ 0 & \boxed{1} & -2 & -1 \\ 0 & 0 & \boxed{2} & 2 \end{pmatrix}$$

We check if any of the starting points at S_1 (other than $[100]'$ of the initial configuration $< S_1, [100]' >$) belong to $R([100]', A_1)$. Since one of the added starting points is also $[100]'$, we have an intersection between an image set at S_1 and the set $R([100]', A_1)$. Therefore, every reachable configuration is recurrent. We can use Lemma 3 to simplify the sets of starting points. After simplification we get the set $\{([100]', A_1), ([052]', A_1), ([000]', A_1)\}$ at S_1. As an example of the test for membership, let us check if $< S_1, [175]' >$ is reachable. Taking the difference between $[175]'$ and each initial point at S_1, we get $[075]'$, $[123]'$, and $[175]'$ respectively. Pre-multiplying $[075]'$ by T we obtain $[7 - 24]'$. Each entry of this vector is divisible by the corresponding boxed diagonal entry of A^*. Therefore, this difference is generated by the matrix A_1. In fact, $A_1[-5220]' = [075]'$. Therefore, this configuration is reachable.

5 Conclusion and Future Work

A formal model of Extended Finite State Machines has been defined. It is suggested that a stabilization technique may be used to simplify the analysis of some classes of EFSMs. A special class of this model called Modular Vector Addition Systems, with or without constant-assigning transitions, is described and analyzed. An algorithm to compute the sets of reachable and recurrent configurations was described. Using the theory of Linear Congruences, a simple test for membership in these sets is developed. The entire process is illustrated with an example. There are many interesting avenues for future work. It would be useful to enlarge the class of update functions u that can be analyzed — for example, transitions that assign constant values to a few variables only. Another interesting problem, with applications to test engineering, is that of finding an *actual* path to reach the final configuration from the initial configuration. While the upper-triangular system described in the appendix can be used to give solutions, it is not necessarily an *optimal* solution based on, say, path length. This is an interesting optimization problem with practical applications. We are currently working on incorporating the results described here in the software implementation described in [6].

Acknowledgements The author would like to thank R.J. Holt, R.P. Kurshan, and anonymous referees for comments and suggestions that have improved the presentation.

References

1. ESTELLE - A Formal Description Technique based on an Extended State Transition Model. ISO **9074**.
2. M.H. Sherif and A.S. Krishnakumar: Performance Evaluation From Formal Specifications of Protocols: A Case Study with LAPD. Proc. of Globecomm'90.
3. A. Bouajjani, J.-C. Fernandez, and N. Halbwachs: Minimal Model Generation. Proc. of II Workshop on Computer-Aided Verification, DIMACS Series Vol **3**, ACM-AMS, 1990, pp. 85-91.

4. A. Bouajjani, J.-C. Fernandez, N. Halbwachs, P. Raymond, and C. Ratel: Minimal State Graph Generation. IMAG Grenoble preprint, 1991.
5. D. Lee and M. Yannakakis: Online Minimization of Transition Systems. Proc. of 24^{th} ACM Symposium on the Theory of Computing, May 1992, pp. 264-274.
6. K.-T. Cheng and A.S. Krishnakumar: Automatic Functional Test generation Using The Extended Finite State Machine Model. Proc. of DAC '93.
7. E.W. Mayr: An Algorithm for the General Petri Net Reachability Problem. SIAM J. Comput. **13**(3), pp. 441-460.
8. B.M. Stewart: Theory of Numbers. The MacMillan Company, New York, 1962.

Appendix

In this appendix, we develop a test for membership in the set $R(\mathbf{x}_0, A)$. The set of integers will be denoted by $\mathbf{Z}$, while $\mathbf{Z}_m$ will refer to the set $\{0, 1, \ldots, m-1\}$. The *greatest common divisor, gcd,* of a set $\{a_1, \ldots, a_n\}$ will be denoted by $(a_1, \ldots, a_n)$. A scalar will be denoted by x, while $\mathbf{x}$ will denote a vector. The i^{th} component of $\mathbf{x}$ will be denoted by x_i. Note that $\mathbf{x}_i$ and x_i are different entities.

A Membership Test

A.1 Linear Congruences

The equation

$$\sum_{i=1}^{p} x_i a_i \equiv b \ mod \ m \tag{3}$$

is a linear congruence. Here a_i and $b \in \mathbf{Z}_m$ and $x_i \in \mathbf{Z}$. For this equation to have a solution for x_i, it is necessary and sufficient that $d = (a_1, \ldots a_p, m)$ divide b. We use the notation $d \mid b$ to indicate that d divides b. We are interested in systems of simultaneous congruences of the form

$$L = \{L_i \mid L_i : \sum_{j=1}^{p} \alpha_{ij} x_j - c_i \equiv 0 \ mod \ m, i = 1, \ldots, n\}.$$

L_i refers to the i^{th} equation in the system L. Two systems of linear congruences L and L' will be called *equivalent* if they have the same solution set. The following lemma provides a key result used in deriving systems equivalent to a given system L.

Lemma 5. *Given L, derive L' as follows: $L_j' = L_j$ when $j \neq k$ and $L_k = \sum_{i=1}^{n} \gamma_i L_i$, where $\gamma_i \in \mathbf{Z}_m$. If $(\gamma_k, m) = 1$, then the system $L_i' \equiv 0 \bmod m, i = 1, \ldots n$ is equivalent to the system $L_i \equiv 0 \bmod m, i = 1, \ldots n$.*

Proof: See Section 4, Chapter 19 of [8]. ■

Consider an MVAS with n variables and whose $G(V)$ has p simple cycles. The result of traversing an arbitrary cycle of G, starting with a vector x_0, can be written as $x_f \equiv (x_0 + Ay) \ mod \ m$. Here x_f is the value of the vector after

traversing the cycle, A is an $n \times p$ matrix, whose columns are the cycle vectors of G. This relation can be written as $Ay \equiv (x_f - x_0)\ mod\ m$. Thus, x_f is reachable from x_0 iff this system has a solution. The set of $b(= (x_f - x_0))$ for which this system has a solution is considered next.

A.2 Characterization of b

We will now derive necessary and sufficient conditions on $\mathbf{b}$ such that $A\mathbf{x} \equiv \mathbf{b}\ mod\ m$ has a solution. By applying a Gaussian elimination algorithm (modified to account for the fact that we are working in $\mathbf{Z}_m$) to this system, we arrive at a system $A^*\mathbf{x}^* \equiv \mathbf{b}^*\ mod\ m$ equivalent to $A\mathbf{x} \equiv \mathbf{b}\ mod\ m$. A^* has the following structure:

$$A^* = \begin{pmatrix} a^*_{11} & * & \dots & * & \vert \dots \\ 0 & a^*_{22} & \dots & * & \vert \dots \\ . & . & \ddots & . & \vert\ A' \\ 0 & . & \dots & a^*_{nn} & \vert \dots \\ & & & & \end{pmatrix}$$

Here, $(a^*_{ii}, m) = (a^*_{ii}, a^*_{i,i+1}, \dots, a^*_{ip}, m) = \delta_i, i = 1, \dots, n$, and entries marked "*" indicate possibly non-zero values. This fact is guaranteed by the elimination algorithm. We now have the following

Theorem 6. *With $A^*, \mathbf{b}^*$, and $\delta_i, i = 1, \dots, n$ as defined above, the system $A\mathbf{x} \equiv \mathbf{b}$ mod m has a solution iff $\delta_i \mid b^*_i, i = 1, \dots, n$.*

Proof: Since the two systems are equivalent, if $\mathbf{b}$ is solvable with A, then $\mathbf{b}^*$ is solvable with A^*. Looking at the i^{th} row of A^*, it follows immediately that $\delta_i \mid b^*_i$, since δ_i is the gcd of all the elements in that row and m. Thus the condition is *necessary*. Now suppose that $\delta_i \mid b^*_i, i = 1, \dots, n$. We now construct a solution for this case. Set $x_j = 0$ for $j = n+1, \dots, p$. Then in the resulting system, $x_n a^*_{nn} \equiv b^*_n\ mod\ m$. But, by hypothesis, $(a^*_{nn}, m) = \delta_i \mid b^*_n$ and hence we can solve for x_n. Let us assume that all $x_i, i = k+1, \dots, n$ have been solved. Consider the k^{th} equation

$$x_k a^*_{kk} + x_{k+1} a^*_{k,k+1} + \dots + x_n a^*_{kn} \equiv b^*_n\ mod\ m \tag{4}$$

Substituting for all the variables $x_i, i = k+1, \dots, n$, we get a new equation

$$x_k a^*_{kk} + \alpha \delta_k \equiv b^*_k\ mod\ m, \tag{5}$$

where α is some integer. This follows from the fact that $\delta_k \mid (a^*_{k,k+1}, \dots, a^*_{kn})$. Since $\delta_k \mid b^*_k$ and $\delta_k \mid a^*_{kk}$, this equation can be solved for x_k. Therefore, by induction, we can solve for $x_i, i = 1, \dots, n$. Thus the condition is *sufficient*. Hence the theorem. ■

During the execution of the triangularizing algorithm above, we can save the matrix of row transformations T_i at each step. Let $T = T_{n-1} \cdots T_1$. Then given a vector $\mathbf{b}$, it is solvable with A if $T\mathbf{b}$ is such that $\delta_i \mid (T\mathbf{b})_i$.

This article was processed using the LaTeX macro package with LLNCS style