Computers and Artificial Intelligence, Vol. 6, 1987, No. 3, 209—228

ON DEADLOCK DETECTION IN SYSTEMS OF COMMUNICATING FINITE STATE MACHINES

Mohamed G. GOUDA

Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712, USA

Eitan M. GURARI, Ten-Hwang LAI[1/]

Department of Computer and Information Science, Ohio State University, Columbus, OH 43210, USA

Louis E. ROSIER[2/]

Department of Computer Sciences, University of Texas at Austin, Austin TX 78712, USA

Abstract. Consider a system of finite state machines that communicate exclusively by exchanging messages over one-directional, FIFO channels. In this paper, we discuss the complexity of detecting deadlocks in such systems: we establish sharp boundaries between the decidable and undecidable cases, and provide complete complexity results for most of the decidable cases. We also give similar results for the problem of detecting unboundedness in such systems. Although in this paper we discuss only the detection of these two design errors, it can readily be shown that other design errors (e.g. unspecified receptions, nonexecutable receptions, and stable states) can be detected as well using similar techniques.

Об определении взаимоблокировки в системах сообщающихся конечных автоматов
М. Г. Гоуда, Е. М. Гурари, Т.-Х. Лай, Л. Е. Росье

Резюме. Рассматривается система конечных автоматов, которые коммуникуют путем обмена сообщениями по однонаправленным каналам FIFO. Обсуждается сложность

[1/] This research was supported by NSF Grant MCS83-08828.

[2/] This research was supported in part by the University Research Institute, the University of Texas at Austin, and the IBM Corporation.

определения взаимоблокировок в таких системах: определяются четкие границы между решаемыми и нерешаемыми случаями и предоставляются полные наборы результатов большинства решаемых случаев. Также представлены результаты для проблемы определения неограниченности в таких системах. Хотя в статье обсуждается только определение этих двух ошибок проектирования, можно также показать, что другие ошибки проектирования (например, неспецифицированные приемы, необрабатываемые приемы, стабильные состояния) могут быть определены, используя подобные методы.

1. INTRODUCTION

In a system of communicating finite state machines (CFSM's, for short), machines communicate exclusively by exchanging messages via connecting channels. There are two one-directional, FIFO channels between any two machines in the system. Each machine has a finite number of states and state transition rules, and each state transition rule is accompanied by either sending one message to one of the machine's output channels or by receiving one message from one of the machine's input channels. Notice that the sending and receiving operations are asynchronous unlike the synchronous operations of Hoare's CSP [7]; however, it is straightforward to use the asynchronous operations to simulate synchronous ones.

Systems of CFSM's are useful in the modeling [21], analysis [1, 2, 4] and synthesis [5, 26] of communication protocols and distributed systems. Examples of real protocols that can be modeled, analyzed, or synthesized as systems of CFSM's include the alternating-bit protocol [1], the binary-synchronous protocol [3], the call establishment/clear protocol in X.25 [5], and X.21 [25]. In performing these functions systems of CFSM's are better than Petri nets [16], which can perform similar functions, for two reasons:

i. Systems of CFSM's are strictly more powerful than Petri nets. (However, Petri nets can be extended in many ways [16] to achieve the full power of CFSM systems.)

ii. The connecting channels between different machines in a system of CFSM's are implicit or built-in, whereas, in the case of Petri nets, they need to be explicitly defined.

One advantage of modeling a communication protocol by a network of CFSM's is to detect many protocol design errors. Two of these design errors which have been discussed extensively in the literature are unboundedness, and deadlocks (cf. [1, 2, 4, 21, 23, 24, 26]). Although in this paper we discuss only the detection of these two design errors, it can be shown that other design errors (e.g. unspecified receptions, nonexecutable receptions, and stable states [2, 26]) can be detected as well using similar decision procedures. In this paper, we are interested in the complexity of the deadlock detection problem (DDP, for short) for systems of CFSM's. This problem has already received some attention. In particular, it is known that the problem is undecidable for systems of two CFSM's as well as for systems of three CFSM's that have only one unbounded channel [2]. On the other hand, the problem is known to be decidable for some other restricted classes of CFSM's [2, 23]. (The restrictions were in the number of CFSM's, in the capacity of the channels, in the languages allowed for communicating through the channels, and in the kind of states allowed in the CFSM's.) However, the question was left unanswered for many other classes of CFSM's and its complexity has not been derived for some of the classes that have known decidable DDP's. Although in the remainder of this introduction we only mention results concerning the DDP, similar complexity results are obtained (see Section 6) for the unboundedness detection problem (UDP, for short). Earlier known results concerning this problem can be found in [24].

So that we can establish sharp boundaries between the decidable and undecidable cases, we first note that the DDP is undecidable (see Section 3) for systems of

(a) two CFSM's that have no mixed states.

(b) three CFSM's that have no mixed states and in which each channel, except for one, is a one-bounded channel over some tally language, i.e. a language in a_1^* for some message a_1.

On the other hand, for systems of CFSM's the problem is shown (see Section 4) to be

(c) PSPACE-complete if one of the channels is bounded by a linear function of the input size and NLOGSPACE-complete (and hence also solvable in polynomial time) if one of the channels is bounded by some fixed constant. In [2], a partial procedure was given to determine, for a network consisting of an arbitrary number of CFSM's, whether any deadlock states were reachable (as well as other error states). The procedure in [2] was shown to terminate for the class of communication networks consisting of two CFSM's in which the communication was bounded in at least one of the channels. The algorithm can also be modified slightly in order to determine whether the remaining channel is bounded. Thus, for the case of two CFSM's the afore-mentioned design errors can be detected by a decision procedure if one of the two channels is known a priori to be bounded. The time complexity of the procedure was not discussed in [2], however, one can show that it must be a function not only of the size of the network, but also of the known channel bound. Our results then provide both upper and lower bounds for this problem using a different proof technique. One should note that, on the other hand, the PSPACE hardness of the problem for the case where both channels are bounded appeared as one of the main results in [18].

(d) decidable in polynomial space if one of the channels is over a (bounded) language in $a_1^* \ldots a_k^*$ for some messages $(a_1, \ldots, a_k)$ which are given as part of the input. If in addition $a_1, \ldots, a_k$ are fixed messages, then the problem is NLOGSPACE-complete. These results generalize the main result in [23] which shows that DDP is decidable in polynomial time if the two CFSM's exchange a single type of message and have no mixed states. (The proof in [23] can be easily generalized for the case where the CFSM's are allowed to have mixed states. The decidability of the result in [23] already follows from [8].)

Moreover, for systems of an arbitrary number of CFSM's the DDP is shown (see Section 5) to be

(e) PSPACE-complete if all the channels are bounded by a linear function of the input size. If, however, the number of CFSM's and the channels are bounded above by some fixed constant, then the problem is NLOGSPACE-complete. (The decidability of the problem, by exhaustive search, was noticed earlier in [2].)

(f) decidable and in fact exponential space-hard if all the channels are over some bounded languages. (The proof relies on similar known results for the reachability problem for vector addition systems [12—14].)

Although the decision procedures given in this paper work only for very restricted systems of CFSM's, they can (in some cases) be used to prove that a more general system is free from certain design errors. For example, the result mentioned in (d) above can be used to prove (rather than decide) that a general network of two machines M and N that exchange any number of messages is free from the above design errors. First, abstract M and N into machines M_1 and N_1, where M_1 sends only one type of message, by considering each message sent by M (and received by N) as the same. (The communication in the other direction is left as is.) Then use the decision procedure in this paper to prove that M_1 and N_1 are free from the above design errors. Second, abstract M and N into two machines M_2 and N_2, where N_2 sends only one type of message. Then use the decision procedure in this paper to prove that M_2 and N_2 are free from the above design errors. It is straightforward to show that if M_1 and N_1 are free from the above design errors and if M_2 and N_2 are free from these errors, then the original machines M and N are also free from these same errors. The converse need not be true, of course. (This verification methodology, beside the fact that some real protocols, notably the window protocol [22], can be modeled as systems of two machines with one of them sending one type of message, explain our interest in such systems.)

2. SYSTEMS OF CFSM'S

A *communicating finite state machine* (CFSM, for short) M_i is a six-tuple

$$(i, m, Q_i, \Sigma, \delta_i, q_{i0}),$$

where

i and m are natural numbers such that $1 \leq i \leq m$ (i is an index that identifies M_i from the other CFSM's in a system (defined later), and m is the number of CFSM's in the system),
Q_i is a finite set whose elements are called *states*,
Σ is an alphabet whose elements are called *messages*,
δ_i is a relation from $Q_i \times \{R, S\} \times \{1, \ldots, i-1, i+1, \ldots, m\} \times \Sigma$ to Q_i whose elements are *transition rules*, and
q_{i0} is a distinguished state in Q_i called the *initial* state.

A state q of M_i is said to be a *receiving* state if $\delta_i \cap (\{q\} \times \{S\} \times \{1, \ldots, m\} \times \Sigma \times Q_i)$ is empty. On the other hand, it is said to be a *sending* state if $\delta_i \cap (\{q\} \times \{S\} \times \{1, \ldots, m\} \times \Sigma \times Q_i)$ is not empty but $\delta_i \cap (\{q\} \times \{R\} \times \{1, \ldots, m\} \times \Sigma \times Q_i)$ is empty. If, however, q is neither a receiving state nor a sending state, then it is said to be a *mixed* state. (Note that states which have no transition rules are considered receiving states by definition.)

A *configuration of the CFSM* M_i is an m-tuple $(q, w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_m)$, where q is a state of M_i and each w_j, called the *content of the channel from* M_j *to* M_i, is a sequence of messages from Σ (i.e. a string in Σ^*). The configuration is said to be an *initial* configuration if q is the initial state of M_i and each w_j is empty. The configuration is said to be a *deadlocked* configuration if q is a receiving state and each w_j is either empty or has a leftmost message a, such that (q, R, j, a, p) is not in δ_i for any p in Q_i.

A *subpath* of M_i is a sequence $v_1, \ldots, v_t$ of transition rules of M_i such that if v_j is a transition rule from state q_j to state p_j, then v_{j+1} is a transition rule from state $q_{j+1} = p_j$ for $1 \leq j < t$. The subpath is said to be a *path* if v_1 is a transition rule from the initial state of M_i. In what follows n_i will denote the number of states in M_i.

A *system* W *of* m *CFSM's* is an m-tuple $(M_1, \ldots, M_m)$ of CFSM's such that $M_i = (i, m, Q_i, \Sigma, \delta_i, q_{i0})$ for $1 \leq i \leq m$ and $Q_i \cap Q_j = \phi$ if $i \neq j$. A *configuration of* W is an m-tuple $(U_1, \ldots, U_m)$, where U_i is a configuration of M_i for $1 \leq i \leq m$. The configuration is said to be an *initial* configuration of W if U_i is an initial configuration of M_i for $1 \leq i \leq m$. The configuration is said to be a *deadlocked* configuration of W if U_i is a deadlocked configuration of M_i for $1 \leq i \leq m$.

A *move* of a CFSM M_i in system W is a pair $(V, V') = ((U_1, \ldots, U_m), (U'_1, \ldots, U'_m))$ of configurations of W that satisfies either of the following two conditions:

(a) $U_k = U'_k$ for $k \neq i$, and if $U_i = (q, w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_m)$, and $U'_i = (q', w'_1, \ldots, w'_{i-1}, w'_{i+1}, \ldots, w'_m)$, then $w_h = w'_h$ for $h \neq j$, $w_j = aw'_j$ for some message a, and (q, R, j, a, q') is a receiving transition rule of M_i. In such a case, the move of M_i is also said to be a *receiving move* in which M_i *receives* a message from M_j.

(b) if $U_i = (q, w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_m)$, then $U'_i = (q', w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_m)$, and if $U_j = (p, w_1, \ldots, w_{j-1}, w_{j+1}, \ldots, w_m)$, then $U'_j = (p, w'_1, \ldots, w'_{j-1}, w'_{j+1}, \ldots, w'_m)$, where $w_h = w'_h$ for $h \neq i$, $w'_i = w_i a$, and (q, S, j, a, q') is a sending transition rule of M_i. In addition, $U_h = U'_h$ for $h \neq i$ and j. In such a case, the move of M_i is also said to be a *sending move* in which M_i *sends* a message to M_j.

A *subcomputation* of W is a sequence of moves of the form $(V_0, V_1), (V_1, V_2), \ldots, (V_{t-1}, V_t)$ for some $t \geq 0$. (Without loss of generality it is assumed that at any given instant of a subcomputation exactly one CFSM is moving.) The subcomputation is said to be a *computation* if V_0 is an initial configuration. The subcomputation is said to be *deadlocked* if V_t is a deadlocked configuration. Two subcomputations of W are said to be

equivalent if they start and end with the same configurations of W, respectively. W is said to be *deadlock free* if it has no deadlocked computations.

The *deadlock detection problem* (DDP, for short) is the problem of determining for any given system of CFSM's whether or not the system is deadlock free.

If $u_1, \ldots, u_t$ is a sequence of transition rules, then its *projection on* M_i is the sequence obtained from $u_1, \ldots, u_t$ after removing the transition rules that do not belong to M_i. A sequence $u_1, \ldots, u_t$ of transition rules is said to be a *subpath* (or a *path*) of W if each u_j, $1 \leq j \leq t$, belongs to some M_i, $1 \leq i \leq m$, and for $1 \leq i \leq m$ the projection of $u_1, \ldots, u_t$ on M_i is a subpath (or a path) of M_i. A subpath is said to *underline* a subcomputation of W if it is the sequence of transition rules that W uses during the subcomputation.

If in each computation of W the sequence of messages that M_i sends to M_j forms a sentence in a given language L, then it is said that the communication through the channel from M_i to M_j is over L (or just that the channel is over L, if no confusion arises). If L is in $a_1^* \ldots a_k^*$ for some $a_1, \ldots, a_k$, then L is said to be a *k-bounded* language. L is said to be *bounded* if it is k-bounded for some k. A one-bounded language is also called a *tally* language.

If at each instant of each computation of W the content of the channel from M_i to M_j consists of no more than h messages, then the channel is said to be *h-bounded*. The channel is said to be *bounded* if it is h-bounded for some h.

The *unboundedness detection problem* (UDP, for short) is the problem of determining for any given system of CFSM's whether or not there is an unbounded channel in the system.

Throughout the paper it is assumed that a system $W = (M_1, \ldots, M_m)$ of CFSM's is represented by a sequence that consists of the representations of $M_1, \ldots, M_m$, respectively. On the other hand, for the purpose of representation each $M_i = (i, m, Q_i, \Sigma, \delta_i, q_{i0})$ is considered as a finite state automaton $(Q_i, \Sigma_i, \delta_i, q_{i0})$ over the alphabet $\Sigma_i = \{R, S\} \times \times \{1, \ldots, i-1, i+1, \ldots, m\} \times \Sigma$.

3. SYSTEMS OF CFSM'S WITH UNDECIDABLE DDP

The undecidability of the halting problem for Turing machines can be used to show that the DDP is undecidable in general for systems of CFSM's [2]. Intuitively, the proof can be applied for any class of systems that allow a CFSM to communicate over any unbounded language with itself (indirectly through one or more intermediate CFSM's). In order to establish sharp boundaries between the decidable and undecidable cases, we need to consider the following two theorems. Both theorems can easily be shown using standard techniques. (In both cases the proof is a refinement of the one used in [2].)

Theorem 3.1. The DDP is undecidable for systems of two CFSM's that have no mixed states.

In a way, the result in Theorem 3.1 is the sharpest possible result because the DDP is decidable for systems of two CFSM's that have only one unbounded channel or only one channel over an unbounded language (see [2] and Theorems 4.1 and 4.2 below). However, the proof of Theorem 3.1 can be refined to show that such is not the case for systems of three CFSM's.

Theorem 3.2. The DDP is undecidable for systems of three CFSM's even when except for one channel all the other channels are one-bounded over some tally languages. The result also holds when the CFSM's have no mixed states.

4. SYSTEMS OF TWO CFSM'S WITH DECIDABLE DDP

This section begins with two important lemmas. The first lemma shows that the computations of any system of two CFSM's can be synchronized. The second lemma states a complexity result for an automaton that can simulate such synchronized computations.

Definition. A subpath of a system W of two CFSM's M_1 and M_2 is said to be a *synchronized subpath* if it is in $(S_1^* R_2^* S_2 R_1)^* S_1^* (S_2 \cup R_2)^*$, where S_i and R_i denote the set of sending and receiving, respectively, transition rules of M_i, for $i = 1, 2$. A subcomputation of W is said to be a *synchronized subcomputation* if it is underlined by a synchronized subpath. (Note that during a synchronized computation the channel from M_2 to M_1 has at most one message until M_1 executes its last receiving move in the computation.)

Lemma 4.1. Let W be any system of two CFSM's M_1 and M_2. Then each computation of W has an equivalent synchronized computation.

Proof. The proof consists of showing that each subcomputation H of W which starts with an empty channel from M_2 to M_1 has an equivalent synchronized subcomputation H', where H and H' are underlined by paths that have identical projections on M_1 and M_2, respectively. The proof is by induction on the number r of messages that M_1 receives from M_2 in H.

If $r = 0$, then H does not contain receiving moves of M_1. Thus H' can be a computation that starts with all the sending moves of M_1 and only after that goes through the moves of M_2.

If $r > 0$, then the computation H can be written as $H_1 H_{R_1} H_2$, where H_{R_1} consists of the first receiving move of M_1. In such a case, all the moves of M_1 in H_1 are sending moves and therefore they can all precede the moves of M_2 in H_1. Thus, with no loss of generality, it can be assumed that $H = H_{S_1^*} H_{(R_2 \cup S_2)^*} H_{R_1} H_2$, where $H_{S_1^*}$ consists only of sending

moves of M_1 and $H_{(R_2 \cup S_2)^*}$ consists only of moves of M_2. On the other hand, for similar reasons it can also be assumed, with no loss of generality, that all the moves of M_2 that follow its first sending move are in H_2, i.e., that $H = H_{S_1}{}^* H_{R_2}{}^* H_{S_2} H_{R_1} H_2$, where $H_{R_2}{}^*$ consists only of receiving moves of M_2 and H_{S_2} consists of the first sending move of M_2. The result then follows by the inductive assumption because H_2 is a subcomputation that has exactly $r - 1$ receiving moves of M_1 and starts with no message in the input channel of M_1. ■

It follows from Lemma 4.1 that if a system W can reach via some computation a deadlocked configuration, or a configuration where the number of messages in one channel is beyond some value, then W can reach a similar configuration via an equivalent synchronized computation. Therefore, it is sufficient to consider only the synchronized computations of W when trying to detect the existence of deadlocked configurations or when trying to decide whether W is bounded. Such computations have the advantage that they can be simulated without the need to record the content of the input channel of M_1 (= output channel of M_2) because each receiving move of M_1 follows immediately the sending move of M_2 that sent the message that M_1 receives.

In what follows, we construct automata which nondeterministically simulate synchronized computations of systems of CFSM's. The constructed automata will have the property that they accept no input if and only if the simulated systems of CFSM's are deadlock free. The complexity of the emptiness problem for such automata follows from [6] and is stated in the following lemma.

Definition. An *r-counter pushdown machine* (or *r-CPM* for short) is a nondeterministic pushdown automaton that is augmented with r one-reversal bounded counters.

Lemma 4.2. The emptiness problem is decidable in $\mathcal{O}(r^* \log t)$ nondeterministic space for the class of r-CPM's if the pushdown is over a one-letter alphabet, where t denotes the number of transition rules in the given automata.

The proof of the last result consists essentially of showing that if an r-CPM accepts some input, then it also accepts some "short" input. The proof is similar to the standard proof that uses pushdown automata for showing the pumping lemma for context-free languages. The main difference is that here the "pumping" is done in two phases. In the first phase, the subcomputations are removed as long as possible while ignoring their effect on the counters. The second phase consists of adding an appropriate number (which is determined by a "smallest" solution to some system of linear Diophantine equations) of subcomputations of the kind that were removed in the first phase, in order to balance the values in the counters. The interested reader should consult [6] for more details. (Another proof of this result can be found in [9].)

The next lemma is also needed here and it is used for obtaining lower bounds.

Lemma 4.3. The DDP for systems of two CFSM's that have no mixed states is

(a) PSPACE-hard if both channels are bounded by a linear function of the input size.
(b) NLOGSPACE-hard if both channels are one-bounded channels over some tally languages.

Proof. From any given linear bounded automaton T and any given input x a system W of two CFSM's can be constructed such that W is deadlock free if and only if T does not accept x. The construction is similar to the one used for Theorem 3.1. (See [2].) Thus, the result in part (a) follows because the linear-bounded automaton acceptance problem is known to be PSPACE-hard [10] and because the channels of W will never hold a content that is longer than $|x| + 2$.

On the other hand, by [20] the graph reachability problem (i.e. the problem of determining for any given directed graph G and any given pair of nodes u_1 and u_2 of G whether or not there is a path from u_1 to u_2) is NLOGSPACE-hard. The result in (b) then also follows because there is a deterministic logspace bounded Turing machine that can from any given directed graph G and any given nodes u_1 and u_2 of G construct a system W of two CFSM's such that W has a deadlocked computation if and only if there is a path in G from u_1 to u_2. W consists of two CFSM's M_1 and M_2, and is constructed from G as shown in Fig. 4.1.

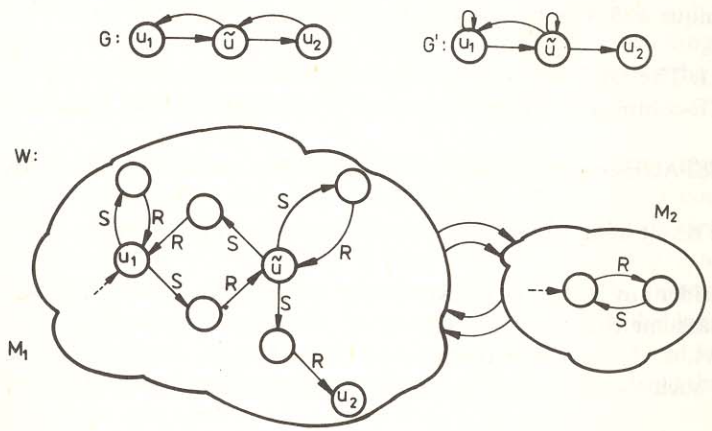

Fig. 4.1. Graphs G and G' and a system W of two CFSM's (S and R denote a sending and a receiving transition rule, respectively).

Let G' be the graph obtained from G by removing all the edges that leave node u_2 and by adding to each of the other nodes a self loop (i.e. and edge from the node to itself). Then M_1 is a CFSM that has a state for each node and a state for each edge of G' has a sending transition rule from state q to state p if and only if q corresponds to some node u of G' and p corresponds to some edge e of G' that leaves u, and has a receiving transition rule from state q to state p if and only if q corresponds to some edge e of G' and p corresponds to the node that e enters in G'. Moreover, the state that corresponds to node u_1 of G' is the initial state of M_1. On the other hand, M_2 consists of a receiving state, which

is also the initial state, and of a sending state. M_2 has a single receiving transition rule that takes M_2 from the receiving state to the sending state, and a single sending transition rule that takes M_2 from the sending state to the receiving state.

Now, each computation of W consists of repeatedly executing the following cycle of moves: a sending move of M_1, followed by a receiving move of M_2, followed by a sending move of M_2, followed by a receiving move of M_1. In addition, at the end of each such cycle M_1 is at a state that corresponds to a node of G' (and hence also of G) and M_2 is in its initial state. Moreover, each state of M_1 that corresponds to a node different than u_2 is a sending state, the state that corresponds to u_2 is a receiving state, and each of these states can be reached by M_1 if and only if the corresponding nodes are reachable from u_1 in G. Thus, W can enter a deadlocked configuration if and only if u_2 is reachable from u_1 in G. ■

By Theorem 3.1 the DDP is undecidable for systems of two CFSM's if no restrictions are made on the communications through the channels. On the other hand, by Theorem 3.2 the DDP is undecidable for systems of three CFSM's even when just one of the channels is unbounded. Thus the following decidable result is the most general that one can expect. The decidability portion of the result was shown earlier in [2] using a different proof technique and with no complexity analysis.

Theorem 4.1. The DDP for systems of two CFSM's is
(a) PSPACE-complete if one of the channels is bounded by a linear function of the input size.
(b) NLOGSPACE-complete if one of the channels is bounded by some fixed constant.

In fact, the result holds also when the CFSM's have no mixed states.

Proof. Consider any system W of two CFSM's M_1 and M_2. In addition, with no loss of generality, assume that the channel from M_1 to M_2 is h-bounded for some h. Then, in what follows, we describe a finite-state automaton M (i.e. a 0-CPM that does not use its pushdown) such that M accepts no input if and only if W is deadlock free.

Essentially, M will nondeterministically simulate (a step at a time) a synchronized computation of W and accept its input string if and only if the simulation terminates at a deadlocked configuration. M will not read its input string during this simulation and thus will either accept all input strings or none at all. In order to simulate a synchronized computation of W, M must be able to record (in its finite state control), at any given instant, the current states of M_1 and M_2, and k pairs of states of M_1, where $k \leq h$. (The recorded pairs of states represent the messages sent by M_1 which have not yet been received by M_2, i.e. the current contents of the h-bounded channel.)

The simulation is done by repeatedly guessing and then simulating each move of M_1 or of M_2 in turn. The only exception is that each receiving move of M_1 is determined simultaneously with the corresponding sending move of M_2 (as long as M_1 is still

receiving messages), thus avoiding the need to record the content of the channel from M_2 to M_1. (M ignores the messages which M_2 sends after M_1 enters a deadlocked configuration, since M_1 cannot receive another message in the simulated computation.)

It is quite straightforward to show that M has at most $n_1 n_2 (n_1^2 + 1)^h$ states (recall that n_i denotes the number of states of M_i for $i = 1, 2$) and that it can be constructed in $\mathcal{O}(h^*(\log n_1 + \log n_2))$ deterministic space. Moreover, from Lemma 4.1 W can reach a deadlocked configuration if and only if it can reach a deadlocked configuration via a synchronized computation. The theorem thus follows from Lemmas 4.2 and 4.3. ■

Theorem 4.1 shows that the DDP is decidable if one channel is known to be bounded. The next result shows that such is also the case if one channel is known to be over a bounded language.

Theorem 4.2. The DDP for systems of two CFSM's is
(a) decidable in polynomial space if one of the two channels is over a bounded language in $a_1^* \ldots a_k^*$ for some messages $(a_1, \ldots, a_k)$ which are given as part of the input.[5]
(b) NLOGSPACE-complete if $a_1, \ldots, a_k$ are fixed messages.

Proof. Consider any system W of two CFSM's M_1 and M_2. In addition, with no loss of generality, assume that the channel from M_1 to M_2 is over a k-bounded language L in $a_1^* \ldots a_k^*$ for some given messages $a_1, \ldots, a_k$. Then in what follows, we describe a $(k-1)$-CPM M such that M accepts no input if and only if W is deadlock free.

M is similar to the finite state automaton constructed in the proof of Theorem 4.1. As was the case in Theorem 4.1, M will nondeterministically simulate a synchronized computation of W and accept its input string if and only if the simulation terminates at a deadlocked configuration. As before, in order for M to simulate a synchronized computation of W, it must be able to record, at any given instant, the current states of M_1 and M_2 as well as the contents of the channel which is over the bounded language (i.e. M_2's input channel). The essential difference in the automaton constructed here is that here M records the content of the channel that is over L in the pushdown and the counters instead of in the finite state control.

Since M differs from the one constructed in the previous theorem only in the way that it records the content of M_2's input channel, our discussion will only concern itself with how the pushdown and counters can be used to store the content of this channel. M starts each computation with empty pushdown and empty counters. During the simulation, if the content of the channel from M_1 to M_2 is empty, then so are the pushdown and the counters of M. If, on the other hand, the content of the channel is $a_i^{j_i} \ldots a_k^{j_k}$ for some $j_i > 0$, then the i'th, ..., $k-1$'st counters hold the values $j_{i+1}, \ldots, j_k$, respectively. The value j_i

[5] It has recently been shown that the nonemptiness problem, for t state r-CPM's, is nondeterministically solvable in time polynomial in r and t. As a result, the problem mentioned here (as well as the one mentioned in Theorem 6.2(b)) can be shown to be NP-complete. See [9].

is stored in the pushdown if $i = 1$ or partially in the pushdown and partially in the $i-1$'st counter if $i > 1$.

When M simulates a move in which M_1 sends a_i it increases the value of the pushdown by 1 if it determines that M_2 received all the messages of type $a_1, \ldots, a_{i-1}$ that were sent earlier. Otherwise, M increases the value of the $i-1$'st counter by 1. Similarly, when M simulates a move in which M_1 receives a message a_i, it decreases the value in the $i-1$'st counter by 1 if $i > 1$ and the counter holds a positive value. Otherwise, M decreases the value of the pushdown by 1.

M can be constructed in deterministic logspace and it has at most kn_1n_2 states and a number of its transition rules which is linear in $kn_1^2n_2^2$. The result now follows from Lemmas 4.1, 4.2, and 4.3(b). ■

We do not know if the polynomial space result in Theorem 4.2 is the sharpest possible result.

5. SYSTEMS OF ARBITRARY NUMBER OF CFSM'S WITH DECIDABLE DDP

It has been noticed in [2] that the DDP is decidable by exhaustive search for systems of CFSM's whose channels are all bounded. The first theorem in this section derives the computational complexity of this problem. Its proof uses the following three lemmas.

Lemma 5.1. Let W be any given system of CFSM's whose channels are all bounded by some given constant h. Then a system W' of CFSM's can be constructed from W and h (deterministically in space that is logarithmic in the length of the representation of W and in the value of h) such that all the channels of W' are bounded by one and W' is deadlock free if and only if W is so. Moreover, if W' has a channel over some nontally language L, then W also has a channel over L.

Proof. Consider any pair M_i and M_j of CFSM's in any system W of CFSM's. Assume that the channel from M_i to M_j is bounded by $h > 1$. Then the communication from M_i to M_j can be modified to go indirectly through some chain of new intermediate CFSM's $M_1', \ldots, M_{h-1}'$ and some new one-bounded channels (see Fig. 5.1). More specifically, each M_j', $1 \le j \le h-1$ can be a CFSM that repeatedly receives a message from its predecessor in the chain, sends an acknowledgement signal to its predecessor, sends the message that it received from its predecessor to its successor, and waits for an acknowledgement signal from its successor.

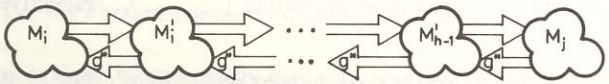

Fig. 5.1.

On the other hand, M_i can be the given CFSM modified so that after sending a message to M_1' it waits for an acknowledgement from M_1' while M_j can be the given CFSM modified so that after receiving a message from M_{h-1}' it sends and acknowledgement to M_{h-1}'. ■

The previous lemma shows how CFSM's can be introduced in order to replace bounded channels with one-bounded channels. The following lemma shows how CFSM's can be introduced in order to replace one-bounded channels over unrestricted languages with one-bounded channels over tally languages.

Lemma 5.2. Let W be any given system of CFSM's whose channels are all bounded by one. Then a system W' of CFSM's can be constructed from W (in deterministic logspace) such that all the channels of W' are one-bounded over some tally languages and W' is deadlock free if and only if W is so.

Proof. Let W be any system of CFSM's and consider any two CFSM's M_i and M_j in W. Assume that the channel from M_i to M_j is one-bounded over some language in $\{a_1, \ldots, a_k\}^*$. Then the communication from M_i to M_j can be modified to go indirectly through some new intermediate CFSM's $M_1', \ldots, M_k'$, where each message a_r, $1 \le r \le k$ is sent from M_i to M_j through M_r' (see also Fig. 5.2). The result now follows because at any given instant of the computation there is at most one message in the channels that are associated with $M_1', \ldots, M_k'$ and therefore M_j must receive the messages that M_i sends in the given order. ■

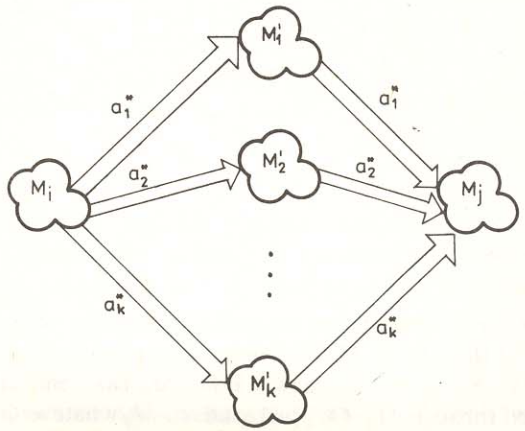

Fig. 5.2.

The next lemma shows how mixed states can be removed from systems of CFSM's.

Lemma 5.3. From any given system W of CFSM's a system W' of CFSM's can be constructed (in deterministic logspace), where W' is deadlock free if and only if W is so,

and the CFSM's of W' have no mixed states. Moreover, if W' has a channel over a nontally language L, or an anbounded channel, or an h-bounded channel for some $h > 1$, then W does so too.

Proof. Let M_i be any CFSM in any system W of CFSM's. Consider any receiving transition rule from any mixed state q of M_i (see also Fig. 5.3). (Note that M_i cannot be deadlocked at state q.) Such a transition rule can be replaced by a sequence of five transition rules that go through four new intermediate states. The sequence consists of a transition rule that sends a signal, say, g to a new CFSM M_{ig}, followed by a transition rule that receives a message g from M_{ig}, followed by a transition rule that receives the message that M_i is supposed to receive in the first place in state q, followed by a transition rule that sends g to M_{ig}, followed by a transition rule that receives a message g from M_{ig}.

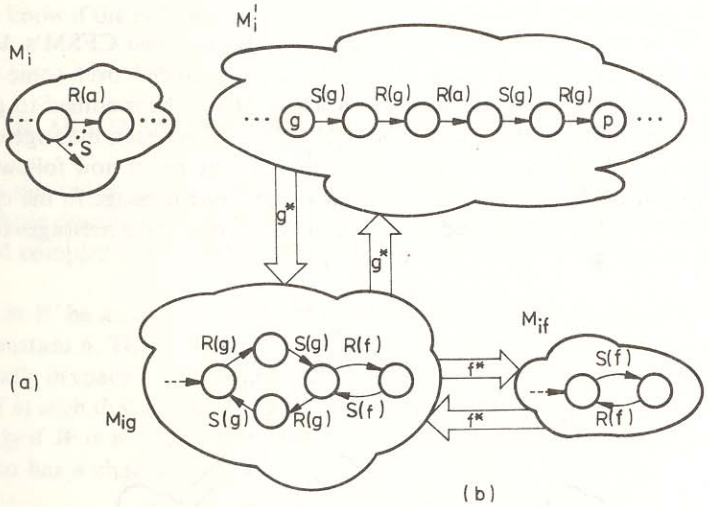

Fig. 5.3. Elimination of mixed states ($R(a)$, $R(g)$, $R(f)$, $S(g)$, and $S(f)$ denote receiving a, receiving g, receiving f, sending g, and sending f, respectively).

M_{ig}, on the other hand, can receive an arbitrary number of g messages from the modified version M_i' of the CFSM M_i. Moreover, when the number of g messages that M_{ig} receives is odd, and only then, a new CFSM M_{if} can send arbitrary number of messages, say f to itself through M_{ig}. (M_{ig} just sends to M_{if} whatever it receives from M_{if}.)

Thus, M_i' can enter a deadlocked configuration on simulating the considered receiving trasition rule of M_i when, and only when, M_{ig} cannot enter such a configuration. Moreover, M_{ig} and M_{if} have no effect on the communications of M_i with the other CFSM's in W. ■

The following theorem can now be shown.

Theorem 5.1. The DDP for systems of CFSM's whose channels are h-bounded is
(a) PSPACE-complete if h is linear in the input size (or if the constant is considered to be an input parameter expressed in unary).
(b) PSPACE-complete if all the channels are over some tally languages and $h = 1$.
(c) NLOGSPACE-complete if h and the number of CFSM's in the systems are no greater than some fixed constant.

Moreover, the results hold also when the CFSM's do not have mixed states.

Proof. (a) and (c) follow from Lemma 5.3 and a proof similar to the one given for Theorem 4.1. (The main difference is that here the contents of all the channels are stored in the finite state control of M and therefore the simulated deadlocked computation of

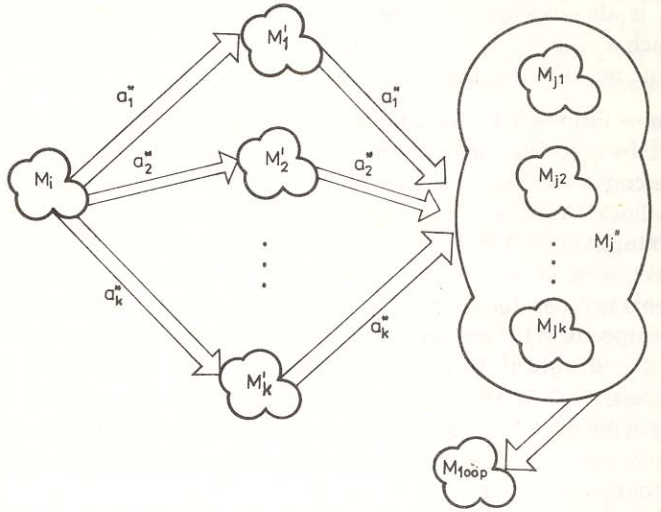

Fig. 5.4.

W need not be synchronized.) (b), on the other hand, follows from (a) and Lemmas 5.1 and 5.2. ■

The next lemma is concerned with the replacement of channels over nontally bounded languages with channels over tally languages. Its proof is a refinement of the one provided for Lemma 5.2. The refinement is needed because the new channels enable the receiving of messages in an order that is different from the one in which they are sent.

Lemma 5.4. Let W be any given system of CFSM's whose channels are over some bounded languages in $a_1^* \ldots a_k^*$ for some known $a_1, \ldots, a_k$. Then a system W' of CFSM's can be constructed from W such that all the channels of W' are over some tally languages and W' is deadlock free if and only if W is so.

Proof. Consider any pair of CFSM's M_i and M_j in any such system W of CFSM's. Assume that the channel from M_i to M_j is over some bounded language in $a_1^* \dots a_k^*$. Then the communication from M_i to M_j can be modified as in the proof of Lemma 5.2 to go indirectly through some new intermediate CFSM's $M_1', \dots, M_k'$, where each message a_r, $1 \le r \le k$, is sent from M_i to M_j through M_r' (see also Fig. 5.2). Here, however, no more than one message can be in the channels that are associated with $M_1', \dots, M_k'$ at any given time or M_j might receive the messages in an order that is different from the one in which they are sent by M_i. This problem can be encountered by modifying M_j so that it enters a deadlock-free computation when such a case arises.

The modified M_j, say M_j'', is a CFSM that consists of k subcomponents $M_{j1}, \dots, M_{jk}$, where each M_{jr}, $1 \le r \le k$, is a slightly modified version of M_j (see Fig. 5.4). More specifically, M_{jr} behaves as M_j as long as it receives messages a_r. On the other hand, upon receiving a message a_t for some $r < t \le k$ the component M_{jr} passes the control to M_{jt}. Moreover, M_{jr} is also capable of receiving a_t in each of its states for each $1 \le t < r$. However, in such a case M_{jr} is forced to an infinite loop in which it repeatedly sends a message to some new CFSM, say, M_{loop}.

The result now follows if all the channels of W are over some bounded languages in $a_1^* \dots a_k^*$ because by modifications similar to those described above a new system W' of CFSM's can be constructed such that all the channels of W' are over some tally languages and W' is deadlock free if and only if W is so. Each computation of W' is either a "successful" simulation of W or an "unsuccessful" simulation. In a "successful" simulation of W by W', each M_j'' receives all the messages a_r, $1 \le r \le k$, that M_i'' sends, before starting to receive any message a_t for some $t > r$ (M_i'' and M_j'' denote the modified versions of M_i and M_j, respectively). Thus, in a "successful" simulation W' reaches a deadlocked configuration if and only if W does so. Moreover, each computation of W can be simulated "successfully" by W'. On the other hand, each "unsuccessful" simulation of W' is caused by some M_{jr} which receives a message a_t before receiving all messages a_r, for some r and t such that $r < t$. Therefore, none of the "unsuccessful" simulations can reach a deadlocked configuration. ■

The decidability [12, 14] of reachability problem for vector addition systems with states [8] and Lemma 5.4 are applied in the proof of the following result.

Definitions. A *vector addition system with states* (*VASS*, for short) is a pair $(u_0, T)_r$, where u_0 is in Z_+^r and T is a finite state automaton over some alphabet in Z^r. (Z denotes the set of integers and Z_+ denotes the set of nonnegative integers.) The *reachability problem for VASS's* is the problem of determining for any given VASS $(u_0, T)_r$ and any given u in Z_+^r whether or not there is some input $u_1 \dots u_t$ such that T accepts it and $u = \sum_{i=0}^{t} u_i$, and $\sum_{i=0}^{j} u_i$ is in Z_+^r for each $0 < j < t$. Each VASS $(u_0, T)_r$ is represented as a sequence that consists of the representation of the integer u_0 followed by a standard representation for the finite state machine T. Unless otherwise specified integers are assigned to be specified in unary.

Theorem 5.2. The DDP is decidable for systems of CFSM's whose channels are over bounded languages in $a_1 \dots a_k$ for some given messages $a_1 \dots a_k$.

Proof. By Lemma 5.4 it is sufficient to consider only those systems whose channels are over tally languages. Thus consider any such system W of m CFSM's $M_1, \dots, M_m$. Let z be the $m(m-1)$ tuple $(0, \dots, 0)$. In what follows, we describe how to construct a VASS $(z, T)_{m(m-1)}$ that, in some sense, will simulate W. The constructed VASS will have the property that it can reach the vector z if and only if W can reach a deadlocked state. Recall that z is reachable in $(z, T)_{m(m-1)}$ only if the associated input of T is *accepted*.

Each symbol in the alphabet of T is an $m(m-1)$ tuple of the form $(0, \dots, 0, d, 0, \dots, 0)$, where $d = -1$ or 1. Such a tuple with $d = -1$ or $d = 1$ in position $(i-1)(m-1) + j$ either corresponds to a message of M_i being received by M_j or to a message of M_i being received by M_j or to a message of M_i being sent to M_j, respectively. On any given input $u_1, \dots, u_t$ the finite state automaton T determines whether or not W has a path that corresponds to the given input. T does so by recording the states of $M_1, \dots, M_m$ in its finite state control and nondeterministically exploring a path (one move at a time) of W whose effect on the channels is represented by some prefix $u_1 \dots u_{t'}$ of $u_1 \dots u_t$, i.e. the sequence of moves on the path must correspond to the sequence of actions on the channels indicated by $u_1 \dots u_{t'}$. (Note that the value of t' is also determined nondeterministically.) T then accepts the input if and only if it determines that the states $q_1, \dots, q_m$ of $M_1, \dots, M_m$, respectively, reached after reading $u_1 \dots u_{t'}$ are all receiving states and that $u_{t'+1} \dots u_t$ correspond only to receiving moves that cannot be used from states $q_1, \dots, q_m$.

Now, if W has a deadlocked computation that ends in the receiving states $q_1, \dots, q_m$ of $M_1, \dots, M_m$, respectively, then there is a t' such that $u_1 \dots u_{t'}$ corresponds to the path taken in this computation. In such a case, all the channels that can provide messages for moves from $q_1, \dots, q_m$ are empty. However, the other channels need not be empty and therefore the need for $u_{t'+1} \dots u_t$. (They serve the purpose of being able to reach z under these circumstances.) On the other hand, if z is reachable in the constructed VASS, then by construction there must be some $u_1 \dots u_t$, which corresponds to a deadlocked computation of W. The result thus follows from [12, 14]. ■

Although explicit upper bounds are known [14], no primitive recursive upper bound is currently known on the time needed for solving the reachability problem for VASS's and thus such is also not implied for the DDP considered in Theorem 5.2. On the other hand, the reachability problem for VASS's is known to be exponential space hard even for the case where integers are given in unary [13, 15]. In what follows, we show that this lower bound be used to show a similar lower bound for systems of CFSM's whose channels are over tally languages.

Theorem 5.3. The DDP requires at least exponential space for systems of CFSM's whose channels are over some tally languages.

Proof. Consider any VASS $(u_0, T)_m$ and any vector u in Z_+^m (all the numbers are assumed to be represented in unary). Then a system W of $m+2$ CFSM's $M_0, \dots, M_m, M_{loop}$ can be constructed such that W is deadlock free if and only if u is not reachable in $(u_0, T)_m$. Furthermore, the size of W will be proportional to the sum of the sizes of $(u_0, T)_m$ and u.

$M_1, \ldots, M_m$ are used only for the purpose of sending back to M_0 the messages that they receive from M_0. Hence, the size of M_i, $1 \leq i \leq m$, is fixed, and does not depend on the size of $(u_0, T)_m$ or u. M_{loop} essentially has no moves and hence its size is also fixed. The construction of M_0, on the other hand, depends very much on $(u_0, T)_m$ and u. M's size therefore will be proportional to that of $(u_0, T)_m$ and u. M_0 starts each computation by sending to M_i, $1 \leq i \leq m$, a number of messages that is equal to the value of the i'th component of u_0. Then M_0 simulates zero or more moves of T. In simulating a move of T, which causes the addition of say vector v, the CFSM M_0 sends to $M_i d$, or receives from $M_i - d$, messages if d is the i'th component of v and $d > 0$ or $d < 0$, respectively, $1 \leq i \leq m$. The encoding of this portion of the simulation forces the size of M_0 to be proportionally as large as the sum of the absolute values of each vector in T. Since the numbers represented in these vectors are encoded in unary, the size of M_0 is proportional to the size of $(u_0, T)_m$. After the simulation of all such moves is completed, M_0 receives from M_i, $1 \leq i \leq m$, a number of messages that is equal to the i'th component in u, and then M_0 enters a distinguished state q. (Note that the time at which the simulation of T moves is over is essentially chosen nondeterministically by M_0.) The encoding of this portion of the simulation causes the size of M_0 to also be proportional to the size of u.

Now, in each state, except for q, M_0 can send an arbitrary number of messages to M_{loop}. On the other hand, from state q, M_0 by receiving a message from either of $M_1, \ldots, M_m$ enters a distinguished sending state which then sends an arbitrary number of messages to M_{loop}. Thus M_0 can enter a deadlocked configuration if and only if it can reach q with all the channels empty.

The details of W's construction are reasonably straightforward and are left to the reader. However, it should be clear that the construction can be carried out in deterministic logspace. The result now follows from [13] (see also [15]). ■

6. SYSTEMS OF CFSM'S WITH UNDECIDABLE/DECIDABLE UDP

Proofs analogous to those utilized in Sections 3—5 can also be used in order to show similar results for the UDP. More explicitly, the UDP is undecidable for the systems that are specified in the following theorem.

Theorem 6.1. The UDP is undecidable for systems of
(a) two CFSM's.
(b) three CFSM's even when except for one channel all the other channels are one-bounded over some tally languages.

The results hold also when the CFSM's have no mixed states.

On the other hand, the UDP is decidable for the systems of two CFSM's that are specified in the following theorem.

Theorem 6.2. The UDP for systems of two CFSM's is
(a) PSPACE-complete if one of the channels is bounded by a linear function of the input size.
(b) decidable in polynomial space if one of the two channels is over a bounded language in $a_1^* \ldots a_k^*$ for some messages $(a_1, \ldots, a_k)$ given as part of the input.
(c) NLOGSPACE-complete if one of the channels is either bounded by some fixed constant or is over a bounded language $a_1^* \ldots a_k^*$ for some fixed messages $a_1, \ldots, a_k$.

Proof. Consider any system W of two CFSM's M_1 and M_2. Then by Lemma 4.1 any computation of W has an equivalent synchronized computation. Therefore it is sufficient to consider ony the synchronized computations of W in order to decide whether or not its channels are bounded.

Thus if W satisfies either of (a), (b), or (c), then a CPM M, similar to the one constructed in the proof of Theorem 4.1 or 4.2, depending on the case, can also be constructed here to simulate the synchronized computations of W. The main difference is that here when M simulates a synchronized computation $H = H_1 H_2$, such that H_1 is in $(S_1^* R_2^* S_2 R_1)^* S_1^*$ and H_2 is in $(S_2 \cup R_2)^*$, it does not totally ignore the sending moves in H_2. Instead, M increases a new distinguished counter by 1 for each sending move of M_2 in H_2. By doing so M, at each instance of the simulation, can use its counters and finite state control to determine the number of messages that are in the channels of W. In particular, M can use the number of these messages at the end of the simulation in order to determine whether or not it equals the input length and accordingly to determine whether to accept or reject the input, respectively.

By construction, W has only bounded channels if and only if M accepts a finite set. The result then follows because the proof of Lemma 4.2 can be generalized to hold also for the finiteness problem (in a manner comparable to the corresponding proofs for finite state automata). ■

Finally, for systems of arbitrary number of CFSM's, the following result holds.

Theorem 6.3. The UDP is decidable in exponential space for systems of CFSM's whose channels are over bounded languages in $a_1^* \ldots a_k^*$ for some given messages $a_1, \ldots, a_k$. Moreover, the problem requires at least exponential space even when $k = 1$. However, if the number of CFSM's in the system is fixed, then the problem is solvable in NLOGSPACE.

Proof. The result follows from proofs similar to those of Theorems 5.2 and 5.3. The hardness result follows from [13, 15], while the upper bounds are derived from [16, 19]. ■

REFERENCES

[1] Bochmann, G.: Finite state description of communication protocols. Computer Networks, Vol. 2, 1978, pp. 361—371.
[2] Brand, D.—Zafiropulo, P.: On communicating finite-state machines. J. ACM, Vol. 30, No. 2, April 1983, pp. 323—342.
[3] Chow, C.—Gouda, M.—Lam, S.: A discipline for constructing multiphase communication protocols. ACM Trans. on Computing Systems, Vol. 3, No. 4, November 1985, pp. 315—343.

[4] CUNHA, P.—MAIBAUM, T.: A synchronization calculus for message-oriented programming. Proc. 2nd International Conf. on Distributed Computing Systems, April 1981, pp. 443—445.

[5] GOUDA, M.—YU, Y.: Synthesis of communicating finite state machines with guaranteed progress. IEEE Trans. on Comm., Vol. COM-32, No. 7, July 1984, pp. 779—788.

[6] GURARI, E.: Transducers with Decidable Equivalence Problem, Univ. of Wisconsin at Milwalkee, Dept. of Computer Science, 1979. Revised, SUNY at Buffalo, Dept. of Computer Science, 1982.

[7] HOARE, C.: Communicating sequential processes. Communications of the ACM, Vol. 21, No. 8, August 1978, pp. 666—677.

[8] HOPCROFT, J.—PANSIOT, J.: On the reachability problem for 5-dimensional vector addition systems. Theor. Comput. Science, Vol. 8, pp. 135—159, 1979.

[9] HOWELL, R.—ROSIER, L.: An analysis of the nonemptiness problem for classes of reversal--bounded multicounter machines. Mathematical Foundations of Computer Science, LNCS 233, August 1986, pp. 422—430. (Also, to appear in J. of Computer and System Sciences).

[10] KARP, R.: Reducibility among combinatorial problems. In: R. Miller and J. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York 1972, pp. 85—103.

[11] KARP, R.—MILLER, R.: Parallel program schemata. J. of Computer and System Sciences, Vol. 3, No. 2, 1969, pp. 147—195.

[12] KOSARAJU, S.: Decidability of reachability in vector addition systems. Proc. of the 14th Annual ACM Symp. on the Theory of Computing, 1982, pp. 267—281.

[13] LIPTON, L.: The Reachability Problem and the Boundedness Problem for Petri Nets are Exponential-Space Hard. In: Conference on Petri Nets and Related Methods, M.I.T., 1975. (Also in Yale Research Report #62 (1976)).

[14] MAYR, E.: An algorithm for the general Petri net reachability problem. SIAM J. of Computing, Vol. 13, No. 3, August 1984, pp. 441—460.

[15] MAYR, E.—MEYER, A.: The complexity of the word problems for commutative semigroups and polynomial ideals. Advances in Mathematics, Vol. 46, Dec. 1982, pp. 305—329.

[16] PETERSON, J.: Petri Net Theory and the Modeling of Systems. Prentice-Hall Inc., Englewood Cliffs, NJ, 1981,

[17] RACKOFF, C.: The covering and boundedness problems for vector addition systems. Theor. Comput. Science, Vol. 6, 1978, pp. 223—231.

[18] RAEUCHLE, T.—TOUEG, S.: Exposure to deadlock for communicating processes is hard to detect. Information Processing Letters, Vol. 21, No. 2, August 1985, pp. 63—68.

[19] ROSIER, L.—YEN, H.: A multiparameter analysis of the boundedness problem for vector addition systems. J. of Computer and System Sciences, Vol. 32, No. 1, 1986, pp. 105—135.

[20] SAVITCH, W.: Relationships between nondeterministic and deterministic tape complexities. J. of Computer and System Sciences, Vol. 4, No. 2, 1970, pp. 177—192.

[21] SUNSHINE, C.: Formal Modeling of Communication Protocols, USC/Inform. Sc. Institute. Research Report 81—89, March 1981.

[22] TANENBAUM, A.: Computer Networks. Prentice-Hall Inc. Englewood Cliffs, NJ, 1981,

[23] YU, Y.—GOUDA, M.: Deadlock detection for a class of communicating finite state machines. IEEE Trans. on Comm., Vol. COM-30, No. 12, December 1982, pp. 2514—2518.

[24] YU, Y.—GOUDA, M.: Unboundedness detection for a class of communicating finite state machines. Information Processing Letters, Vol. 17, December 1983, pp. 235—240.

[25] WEST, C.—ZAFIROPULO, P.: Automated validation of a communications protocol: The CCITT X.21. IBM Journal of Research and Development, Vol. 2, January 1978, pp. 60—71.

[26] ZAFIROPULO, P., ET. AL.: Towards analyzing and synthesizing protocols. IEEE Trans. on Comm., Vol. COM-28, No. 4, April 1980, pp. 651—661.

Received April 5, 1986 — revised version September 26, 1986