A Unified Approach to the Deadlock Detection Problem in Networks of Communicating Finite State Machines

WUXU PENG
Dept of Computer Science
Southwest Texas State University
San Marcos, TX 78666

S. PURUSHOTHAMAN
Department of Computer Science
The Pennsylvania State University
University Park, PA 16802

Abstract

We consider the deadlock detection problem (DDP) of networks of communicating finite state machines (NCFSMs). The DDP problem is known to be undecidable for NCFSMs. In this paper, we provide a characterization of those subclasses of networks for which the deadlock problem is decidable. We also provide a proof technique based on our characterization and illustrate our technique on an example.

1 Introduction

Communicating finite state machines are a very useful abstract model for specifying, verifying and synthesizing communication protocols [2, 3]. In this model a system of communicating finite state machines can communicate typed messages asynchronously with each other over uni-directional, unbounded FIFO channels.

A central issue in this model is whether a network of communicating finite state machines (NCFSMs) is free of progress errors. Several widely addressed progress properties are: freedom from deadlocks, freedom from unspecified receptions, and freedom from unbounded communication. The problem of checking for non-progress in NCFSMs is known to be undecidable [1, 2, 3]. Because of this negative conclusion a natural question is: "For what classes of NCFSMs is the progress problem decidable?" A large amount of literatures (e.g. [8, 6, 7]) has been devoted to identifying classes of NCFSMs for which some of the progress problems are decidable. Specific classes of NCFSMs are usually obtained by placing restrictions on the structure of the systems. For instance, the number of machines in the system, the number of message types allowed, and channel capacity (maximum number of pending messages allowed in channels) etc. However the underlying question

For what classes of NCFSMs are the progress problems decidable?

has not been answered before.

In this paper, we investigate the deadlock detection problem from the language-theoretic point of view. Techniques developed here are applicable to other progress problems. Specifically, we give a necessary and sufficient condition on execution sequences of a class of network

such that the deadlock detection problem is decidable for that class of NCFSMs. We believe that our work provides insight into the underlying nature of decidability of deadlock detection problem. We provide an example to illustrate our technique.

This paper is organized as follows: In Section 2 we introduce necessary notations and definitions, we present our main result in Section 3, in Section 4 we give an example to illustrate the main result and conclude in Section 5.

2 Preliminaries

A communicating finite state machine (CFSM) is a labeled directed graph with a distinguished initial state, where each edge is labeled by an event. The events of a CFSM are **send** and **receive** commands over a finite set of message types M. The communication between CFSMs is assumed to be asynchronous (i.e., non-blocking sends and blocking receives). Consequently, we assume the availability of an infinite buffer between each pair of machines to store pending messages.

Let $I = \{1, \cdots, n\}$, where $n \geq 2$ is some constant. Formally, we have

Definition 2.1 *(CFSM) A CFSM P_i is a four-tuple $(S_i, \langle M_{i,j}\rangle_{j\in I} \cup \langle M_{j,i}\rangle_{j\in I}, \delta_i, p_{0i})$, where*

(1) *S_i is the set of local states,*

(2) *$M_{i,j}$ is the set of message types that P_i can send to machine P_j, and $M_{j,i}$ is the set of message types that P_i can receive from machine P_j. It is assumed that $M_{i,i} = \emptyset$, since P_i can not directly send messages to or receive messages from itself.*

(3) *Let $-M_{i,j} = \{-m | m \in M_{i,j}\}$ and $+M_{j,i} = \{+m | m \in M_{j,i}\}$.*
δ_i is a partial mapping, δ_i: $S_i \times (\langle -M_{i,j}\rangle_{j\in I} \cup \langle +M_{j,i}\rangle_{j\in I}) \times I \longrightarrow 2^{S_i}$. $\delta_i(p, -m, j)$ is the set of new states that machine P_i can possibly enter after sending message of type m to machine P_j, and $\delta_i(p, +m, j)$ is the set of new states that machine P_i can possibly enter after receiving message of type m from machine P_j.

(4) *p_{0i} is the initial local state.*

A state p in P_i is said to be a send (receive, resp.) state iff all of its outgoing edges are send (receive, resp.) edges. p is said to be a mixed state iff it has both outgoing send and receive edges. Let $RMsg(p)$ be the set of message types that can be received in state p, i.e. $RMsg(p) = \{m | \exists p' \ \exists j \ p' \in \delta_i(p, +m, j)\}$. Define

$$
\begin{array}{ll}
M_i = \cup_{j\in I}(M_{i,j} \cup M_{j,i}), & \\
M = \cup_{i\in I} M_i, \text{ and} & \\
-M_i = \cup_{j\in I} - M_{i,j}, & +M_i = \cup_{j\in I} + M_{j,i}, \\
\pm M_i = -M_i \cup +M_i, & \\
-M = \cup_{i\in I} - M_i, & +M = \cup_{i\in I} + M_i, \\
\pm M = +M \cup -M. &
\end{array}
$$

Without loss of generality, we assume that $M_{i,j} \cap M_{k,l} = \emptyset$ if $(i,j) \neq (k,l)$. Due to this assumption, for any $a \in \pm M_i$ we can simplify the notation $\delta_i(p,a,j)$ to $\delta_i(p,a)$.

Definition 2.2 (*Network of communicating finite state machines*) *A network of communicating finite state machines (NCFSM) is a tuple $N = \langle P_1, \cdots, P_n \rangle$, where each P_i $(i \in I)$ is a CFSM.*

A global state of N is a tuple $[\langle p_i \rangle_{i \in I}, \langle c_{i,j} \rangle_{i,j \in I}]$, where p_i is a local state of machine P_i, $c_{i,j}$ is the sequence of messages in the channel from machine P_i to P_j. Let V be the cartesian-product of the sets $S_1, \cdots, S_n$, i.e. $V = S_1 \times \cdots \times S_n$, and let C be the cartesian-product of the sets $M^*_{1,2}, \cdots, M^*_{1,n}, M^*_{2,1}, \cdots, M^*_{n,n-1}$.

Initially, N is in its initial state $[\langle p_{0i} \rangle_{i \in I}, \langle c_{i,j} \rangle_{i,j \in I}]$, where $c_{i,j} = \varepsilon$ $(i \neq j)$. Let $[\langle p_i \rangle_{i \in I}, \langle c_{i,j} \rangle_{i,j \in I}]$ be a global state. The global state transition function $\delta_N : (V \times C) \times \pm M \longrightarrow 2^{V \times C}$ is a partial function defined as:

(1) if $\exists i, j \in I$ $(i \neq j)$ such that $p'_i \in \delta_i(p_i, -m, j)$ then
$[\langle p'_i \rangle'_{i \in I}, \langle c'_{i,j} \rangle'_{i \in I}] \in \delta_N([\langle p_i \rangle_{i \in I}, \langle c_{i,j} \rangle_{i,j \in I}], -m)$, *where*
$p_k = p'_k$ $(k \neq i)$, $c_{k,l} = c'_{k,l}$ $(k \neq i$ *or* $l \neq j)$, *and* $c'_{i,j} = c_{i,j}.m$.

(2) if $\exists i, j \in I$ $(i \neq j)$ such that $p'_i \in \delta_i(p_i, +m, j)$ then
$[\langle p'_i \rangle'_{i \in I}, \langle c'_{i,j} \rangle'_{i \in I}] \in \delta_N([\langle p_i \rangle_{i \in I}, \langle c_{i,j} \rangle_{i,j \in I}], +m)$, *where*
$c_{k,l} = c'_{k,l}$ $(k \neq j$ *or* $l \neq i)$, *and* $m.c'_{j,i} = c_{j,i}$.

We use $P_i \rightarrow P_j$ to denote the channel from P_i to P_j. In essence, the first case in Definition 2.2 denotes the event that P_i sends a message m to P_j, which causes the message m to be appended to the end of channel $P_i \rightarrow P_j$. The second case represents the event that P_i receives a message of type m sent by P_j, which has the effect of removing the first message (which must be of type m, or error (unspecified reception) would occur) in the channel $P_j \rightarrow P_i$. In both cases, after the successful completion of the event, P_i enters local state p'_i while all other machines remains in the same local states and the contents of all other channels are unchanged.

To simplify the expressions, we will use the notation $[v, c]$ to denote a global state whenever necessary, where by convention $v = \langle p_i \rangle_{i \in I} \in V$, and $c = \langle c_{i,j} \rangle_{i,j \in I} \in C$. $[v_0, c_0]$ will be used to denote the initial state.

Definition 2.3 (*Reachability function*) *Let $N = \langle P_1, \cdots, P_N \rangle$ be an NCFSM. The global state transition function δ_N can be easily extended to the following reachability function δ^*_N : $(V \times C) \times \pm M^* \longrightarrow 2^{V \times C}$,*

(1) $\delta^*_N([v,c], \varepsilon) = \{[v,c]\}$.

(2) $\delta^*_N([v,c], e.a) = \{[v',c'] \mid \exists [v'',c''] \in \delta^*_N([v,c], e),\ [v',c'] \in \delta_N([v'',c''], a)\}$.

We often write $\delta^*_N([v_0, c_0], e)$ as $\delta^*_N(e)$. Furthermore, we will drop the subscripts in δ_N and δ^*_N if no confusion arises.

Given a reachability function δ^*, we define the set of all states reached as a result of some execution to be the *reachability set.* More formally,

Definition 2.4 (*Reachability sets*) *Let $N = \langle P_1, \cdots, P_n \rangle$ be an NCFSM. The reachability set $RS(N)$ is the set of all reachable global states, $RS(N) = \{[v, c] \mid [v, c] \in \delta^*(e), e \in \pm M^*\}$.*

In the rest of this paper sequences of events will form the back bone of our discussions. Hence, we will use the following abbreviations:

Event Sequence A string $e \in \pm M^*$ is an *event sequence.*

Executable An event sequence e is *executable*, notated as $\delta^*(e) \neq \emptyset$, if $\delta^*(e)$ is defined.

Feasible An event sequence $e \in \pm\Sigma^*$ is *feasible*, if [1]

1. $\forall e' \in pref(e)\ \forall g \in \Sigma\ |e'|_{+g} \leq |e'|_{-g}$; and
2. $\forall i, j \in I$, if $+g_{i,j}$ is the k^{th} receive event in $f_{i,j}(e)$, then $-g_{i,j}$ is the k^{th} send event in $f_{i,j}(e)$.

Feasibility is a very strict requirement. We can easily show that feasible sequences are context-sensitive. We will use $F(N)$ to denote the set of all feasible event sequences of a network N.

Stable An event sequence e is *stable*, if it is feasible and in addition it contains the same number of send and receive events of any message type, i.e., $\forall a \in \Sigma \mid e_{+a} \mid = \mid e_{-a} \mid$. We will use $SE(N)$ to denote the set of all stable event sequences of a network N.

It should be clear to the reader that a definition of deadlock (or any progress error) can be stated in terms of execution sequences. Of course, execution sequences capture the semantics (and causality) of processes and buffers present in a network. The causality constraints among possible actions in a network can be split into those imposed by the behavior of FIFO buffers and those imposed by the sequencing constraints of the processes. The notion of feasible and stable (event) sequences defined above capture the constraints of the FIFO buffers in the network. The sequencing constraints of the network can be easily captured as a shuffle of the sequencing constraints of the individual processes. Formally, we define:

Definition 2.5 (*Shuffle-product of NCFSMs*) *Let $N = \langle P_1, \cdots, P_n \rangle$ be an NCFSM. The shuffle-product of N, notated as $SP(N)$, is a four-tuple (V, M, Δ, v_0), where*

(1) $V = S_1 \times S_2 \times \cdots \times S_n$.

(2) $v_0 = [p_{01}, p_{02}, \cdots, p_{0n}] \in V$.

(3) The transition function Δ: $V \times \pm M \longrightarrow 2^V$ *is defined as*

[1] For a string w, $|w|$ is the length of w, $|w|_a$ is the number of occurrence of letter a in w and $pre(w)$ is the set of all prefixes of w.

$v' \in \Delta(v,a)$, *where* $a \in \pm M_i \subseteq \pm M$, *iff* $v'_j = v_j$ $(j \in I \ \&\ j \neq i)$ *and* $v'_i \in \delta_i(v_i,a)$.

The shuffle-product $SP(N)$ can be viewed as a (nondeterministic, in general) finite automaton by identifying some subset of V as final state set. We use $SP(N)(F)$ to notate the finite automaton obtained from the shuffle-product with $F \subseteq V$ as the final state set.

A tuple $v \in V$ is a *receive* node if for each $i \in I$ v_i is a receive state in P_i. $REV(N)$ denotes the set of all receive nodes in V.

A number of progress properties have received wide attention and one of the well known progress properties is the deadlock detection problem (DDP).

Definition 2.6 (*Deadlock*) *Let* $N = \langle P_1, \cdots, P_n \rangle$ *be an NCFSM and* $[v,c] \in RS(N)$ *be a global state.*

$[v,c]$ *is a* deadlock state *if the predicate*

$$d([v,c]):\ v \in REV(N)\ \&\ c = c_0$$

holds.

The network N *is* free of deadlocks, *if the predicate*

$$\forall [v,c] \in RS(N)\ (\textbf{not}\ d([v,c]))$$

holds.

It is well known that in general it is undecidable whether an NCFSM is free of deadlocks ([2, 3]). We state this fact in following theorem.

Theorem 2.1 *DDP is undecidable.*

3 A Unified Approach to DDP

Theorem 2.1 states that the problem of detecting deadlocks in a network is undecidable in general. To cope with this negative result, many special classes of NCFSMs have been identified for which the DDP is decidable. This usually involves finding sufficient conditions under which the problem becomes decidable.

In this section we take a different approach to this problem. Instead of trying to find special classes of NCFSMs with decidable DDP, we give a necessary and sufficient condition under which the DDP of a given class of NCFSM is decidable.

First let us formalize the concept of *classes* of NCFSMs.

A *class* $\mathcal{N}$ of NCFSMs is a tuple (Q, Σ, T), where

1. Σ is a finite (or countably infinite) set of message types,
2. Q is a collection of NCFSMs each of which draws message types from Σ,

3. T is a predicate which characterizes the properties of NCFSMs in the Q. For instance, T may be the predicate: "each $N \in Q$ has only two CFSMs", which is the class of NCFSMs with two CFSMs.

Let $\mathcal{N} = (Q, \Sigma, T)$ be a class of NCFSMs, and let $N = \langle P_1, \cdots, P_n \rangle \in \mathcal{N}$. We say that the DDP is *decidable* for a network class $\mathcal{N}$, if the predicate

$$D(\mathcal{N}) : \forall N \in \mathcal{N}\ \exists [v, c] \in RS(N)\ (v \in RV(N) \& c = c_0).$$

is decidable. In the following we would like to relate the decidability of deadlock to conditions on execution sequences. It is easy to see that if an execution sequence e leads to a deadlock state, then it should have the property that for every send event in e there should be a corresponding receive event. In fact, e should be a stable event sequence.

In general the set $SE(N)$ is context-sensitive, since we can easily construct a linear bounded automaton (LBA) that accepts $SE(N)$. Even for some trivial classes of NCFSMs $SE(N)$ remains context-sensitive. However in order to check if DDP is decidable for N, it is not necessary to know every member in $SE(N)$. This observation is based on the fact that there a number of event sequences that are really interleavings of the same set of actions of the processes, and hence lead to the same global state. We should therefore consider these interleavings as being equivalent. Formally we have, two event sequences e_1 and e_2 are *equivalent*, notated as $e_1 \simeq e_2$, iff

1. e_1 and e_2 are permutations of each other,

2. $\delta^*(e_1) = \delta^*(e_2)$.

Since we are really interested in stable event sequences, we will say two event sequences e_1 and e_2 are *stable equivalent*, notated as $e_1 \simeq_{st} e_2$, iff $e_1 \simeq e_2$ and both e_1 and e_2 are stable.

Define $class(e) = \{e' \mid e' \simeq_{st} e\}$. It is easy to see that $\simeq_{st}$ is an equivalence relation on $SE(N)$ and $class(e)$ is an equivalence class.

As mentioned earlier, we only need a *representative* from each equivalence class of $SE(N)$ to check for existence of deadlock in N. To that end, we define a language $C_N \subseteq SE(N)$ to be a *stable cover set* for N if

$$\forall e \in SE(N)(equiv(e) \cap C \neq \emptyset). \tag{1}$$

With the concepts of stable cover sets and shuffle-product automata, we have the following theorem regarding the decidability of DDP for classes of NCFSMs.

Theorem 3.1 *Let $\mathcal{N} = (Q, \Sigma, T)$ be a class of NCFSMs. Let $\mathcal{R} = \{L(M_v) \mid M_v = SPA(N, \{v\}) : v \in V_N\}$, i.e. $\mathcal{R}$ is the collection of shuffle-product automata, each of which has some node $v \in V_N$ as the single final state. The DDP is decidable for $\mathcal{N}$ if and only if for every network $N \in \mathcal{N}$, there exists a stable cover set $C_N \subseteq SE(N)$ such that for every $M_v \in \mathcal{R}$, the predicate $C_N \cap L(M_v) = \emptyset$ is decidable.*

Proof: Let $\mathcal{N} = (Q, \Sigma, T)$ be a class of NCFSMs.

If. Let N be a member in $\mathcal{N}$, and assume that we have a stable cover set $C_N \subseteq SE(N)$ which satisfies the condition listed in the theorem.

Claim: $[v, c_0]$ is a deadlock state if and only if $C_N \cap L(M_v) \neq \emptyset$.

Proof of the claim:

Assume that $C_N \cap L(M_v) \neq \emptyset$. Since C_N is a stable cover, each $e \in C_N$ is stable. Moreover C_N contains at least one element from each equivalence class $equiv(e)$. As $L(M_v)$ contains all executable event sequences which can lead to a reachable global state of the form $[v, c]$, there must be at least one executable event sequence $e \in C_N \cap L(M_v)$. such that $[v, c_0] \in \delta(e)$.

Conversely assume that $[v, c_0]$ is a reachable deadlock state. Therefore there exists an executable sequence e where $[v, c_0] \in \delta(e)$. As C_N is a stable cover, there must exist $e' \simeq_{st} e$ and $e' \in C$. Obviously $e' \in L(M_v)$, hence $C_N \cap L(M_v) \neq \emptyset$.

By definition DDP is decidable for N. Since DDP is decidable for any $N \in \mathcal{N}$ we conclude that DDP is decidable for the class $\mathcal{N}$.

Only If. Assume that DDP is decidable for $\mathcal{N}$. We must show that for every network $N \in \mathcal{N}$ there exists a stable cover set C_N for N such that the emptiness problem of $C_N \cap L(M_v)$ is decidable for all $v \in V_N$. In the following argument, the symbol $\Longrightarrow$ is used to express logical implication. "$A \Longrightarrow B$" means that the decidability of A implies the decidability of B. We have following logical reasoning:

$$
\begin{aligned}
& D(\mathcal{N}) \\
& \Longrightarrow \forall\, N \in \mathcal{N}\ (\exists\, [v, c] \in RS(N)\ (v \in RV(N)\ \&\ c = c_0)) \\
& \Longrightarrow \forall\, N \in \mathcal{N}\ (\exists\, e \in NL(N)\ \exists\, [v, c] \in \delta(e)\ (v \in RV(N)\ \&\ c = c_0)) \\
& \Longrightarrow \forall\, N \in \mathcal{N}\ (\exists\, e \in SE(N)\ \exists\, [v, c_0] \in \delta(e)) \\
& \Longrightarrow \forall\, N \in \mathcal{N}\ (\exists\, e \in SE(N)\ \exists\, v \in RV(N)\ (e \in L(M_v)) \\
& \Longrightarrow \forall\, N \in \mathcal{N}\ (\exists\, v \in RV(N)\ (e \in SE(N) \cap L(M_v))) \\
& \Longrightarrow \forall\, N \in \mathcal{N}\ (\exists\, v \in RV(N)\ (SE(N) \cap L(M_v) \neq \emptyset)).
\end{aligned}
$$

However, $SE(N)$ itself is a stable cover set.

□

It is obvious that the problem of finding a cover set with the stated properties is undecidable. However Theorem 3.1 presents a unified view of DDP for NCFSMs. It sheds new light on the decidability of DDP for NCFSMs in following sense: Given a specific class $\mathcal{N}$ of NCFSMs, to test if DDP is decidable for $\mathcal{N}$ we try to find a cover set satisfying the theorem. If we can find such a cover, we can conclude that DDP is decidable for $\mathcal{N}$. We shall illustrate this idea in next section.

In the methodology engendered by this characterization, we expect to be able to define the cover set C_N for a network N independent of the transitions (or semantics) of a particular network N. Furthermore, we expect to be able to check that such a cover set has the necessary properties. Based on the fact that the language of a shuffle-product automaton is regular, we have the following:

Corollary 3.1 *For a class of networks $\mathcal{N}$, let $\mathcal{L}$ be a family of languages such that the cover set C_N for every network $N \in \mathcal{N}$ belongs to $\mathcal{L}$. If*

1. *$\mathcal{L}$ is closed under intersection with regular languages, and*
2. *The emptiness problem is decidable for $\mathcal{L}$*

then the deadlock detection problem is decidable for the class of networks $\mathcal{N}$.

The Corollary given above provides a very tight sufficient condition to show that a class of networks has decidable DDP.

4 Applications

An NCFSM $N = \langle P_1, \cdots, P_n \rangle$ is *cyclic* if the topology graph of the network N is a simple cycle. More formally, N is a cyclic network if there exists a permutation $\{i_1, i_2, \cdots, i_n\}$ of the set I such that P_{i_j} can only send message to $P_{i_{j+1}}$ and receiving message from $P_{i_{j-1}}$ ($1 \leq j \leq n$, module $n+1$).

All two-machine networks are cyclic. As it is known that DDP is even undecidable for the class of two-machine networks, so DDP is undecidable for general cyclic networks.

However DDP is decidable for cyclic networks where only one channel is unbounded. Let $\mathcal{N}^1_{1-u-cyc}$ be the class of cyclic networks of which only one channel is unbounded (referred to as 1-U cyclic NCFSMs, for short).

Let $N = \langle P_1, \cdots, P_n \rangle$ be a 1-U cyclic NCFSM. To simplify the discussion we assume without loss of generality that P_i can only send to P_{i+1} and receive from P_{i-1} ($1 \leq i \leq n$, module $n+1$) and only the channel $P_1 \rightarrow P_2$ is unbounded. Let $L = (A^*B)^*H$, where

$A = (\cup_{2 \leq i \leq n-1} - \Sigma_{i,i+1}) \cup (-\Sigma_{n,1}) \cup (\cup_{2 \leq i \leq n-1} + \Sigma_{i,i+1}) \cup (+\Sigma_{n,1})$,

$B = (-\Sigma_{1,2})(+\Sigma_{1,2})$,

$H = (A \cup (-\Sigma_{1,2}))^*$.

The following lemma is a generalization of a lemma from [4].

Lemma 4.1 *Let $N = \langle P_1, \cdots, P_n \rangle$ be a 1-U cyclic NCFSM. For each executable event sequence e there exists another event sequence $e' \simeq e$ and $e' \in L$.*

Proof: Let $N = \langle P_1, \cdots, P_n \rangle$ be a 1-U cyclic NCFSM. The proof is by induction on the number k of receive events from $+\Sigma_{1,2}$.

Basis: $k = 0$. Since e does not contain events from $+\Sigma_{1,2}$, $e \in H$. Clearly the conclusion is true.

Induction: Assume that the conclusion holds for some $k > 0$. Let e be an executable event sequence which contains $k+1$ receive events from $+\Sigma_{1,2}$. Let $s_{1,2}$ be the first send event from $-\Sigma_{1,2}$ and $r_{1,2}$ be the first receive event from $+\Sigma_{1,2}$ in e. We can rewrite e as $e = w_1.s_{1,2}.w_2.r_{1,2}.w_3$, where w_1 and w_2 do not contain any events from $(+\Sigma_{1,2})$.

As the network is cyclic (P_2 can only send to P_3 and receive from P_1), we can move all the send events from $(-\Sigma_{2,3})$ in w_2 before $s_{1,2}$. Let $w = w_1'.s_{1,2}.w_2'.r_{1,2}.w_3$ be the event sequence

after such a reordering of e, where w_2' does not contain any events from $(-\Sigma_{2,3}) \cup (+\Sigma_{1,2})$. It is easy to see that $w \simeq e$. Since w_2' does not contain events initiated by P_2, we can move the event $r_{1,2}$ as the immediate successor of $s_{1,2}$. Let $w' = w_1'.s_{1,2}.r_{1,2}.w_2'.w_3$ be the event sequence after such reordering of w. Still $w' \simeq e$ holds.

Since w_3 contains k receive events from $+\Sigma_{1,2}$, by induction hypothesis there exists another event sequence $w_3' \simeq w_3$ and $w_3' \in L$. Hence $e \simeq e' \in L$, where $e' = w_1'.w_2.s_{1,2}.r_{1,2}.w_3'$.

□

Lemma 4.1 says that for any executable sequence e there is another sequence $e' \simeq e$ such that when e' is executed, there is at most one pending message in the only unbounded channel $P_1 \rightarrow P_2$ as long as P_2 can still receive. Therefore to check for deadlocks, we need only concentrate on the behavior of other bounded channels.

Theorem 4.1 *There exists a regular stable cover set for each network* $N \in \mathcal{N}_{1-u-cyc}$.

Proof: Let $N = \langle P_1, \cdots, P_n \rangle$ be a 1-U cyclic NCFSM.

To find a stable cover set for N, we need to construct a language which contains at least one element from each equivalent class $SE(e)$ where e is a stable event sequence.

Let $L' = (A^*B)^*A^*$, where A, B are defined as above. Notice that each $e \in L'$ contains the same number of events from $-\Sigma_{1,2}$ and $+\Sigma_{1,2}$. We can conclude from Lemma 4.1 that for each stable event sequence e, there exists $e' \simeq e$ and $e' \in L'$.

We can construct a finite state automaton F to accept the set of all stable events in L' as follows. Let each state in F record the number of pending event types, message types and message positions it has seen so far. The only exception is that when F sees an event $-g \in -\Sigma_{1,2}$, it will expect another event $+g \in +\Sigma_{1,2}$ in its next step. Notice that since all the channels except $P_1 \rightarrow P_2$ are bounded and the number of message types each machine can send is finite, the number of states in F is finite. The final state set in F includes only those states in which F has seen a stable event sequence (only the start state need be in the final state set). Therefore $L(F)$, the language accepted by F is a cover set for N.

□

By Corollary 3.1, the DDP is decidable for $\mathcal{N}_{1-u-cyc}$.

5 Conclusions

Deadlock detection problem for networks of communicating finite state machines has been known to be undecidable. The undecidability stems from the fact that even a two-machine NCFSM has the same computing power as a Turing machine. With special restrictions, specific classes of NCFSMs have been found for which the deadlock detection problem is decidable. However the underlying question "For what classes of NCFSMs is the DDP decidable," has not been answered before.

In this paper we considered the deadlock detection problem in NCFSMs from the formal language point of view. We have given a necessary and sufficient condition to show decidability

of detecting deadlocks in classes of NCFSMs. We believe our work reveals the nature of decidability vs. undecidability of DDP.

The language concepts were first (as far as the authors know) introduced in the analysis of NCFSMs by K. Okumura [5]. The work in [5] concentrated mainly on establishing correspondence between the languages and the networks, in analogy with the traditional formal languages theory. However, the executable event sequences are context-sensitive even for many trivial networks. It appears that a direct use of the executable event sequences would not aid the analysis. This was our main motivation for introducing the concept of cover set for a particular property (DDP in this paper). Although we only discussed the decidability of DDP in this paper, the idea can also be applied to the decidability of detecting other properties such as unspecified receptions and unboundedness.

References

[1] G. Bochmann. Finite State Description of Communication Protocols. *Computer Networks*, Vol.2, 1978, pp.361–371.

[2] D. Brand and P. Zafiropulo. On Communicating Finite-state Machines. *JACM*, 30(2), 1983, pp.323–342.

[3] M. Gouda, E. Manning, and Y. T. Yu. On the Progress of Communication between Two Finite State Machines. *Information and Control*, 63(3), 1984, pp.308–320.

[4] M. Gouda, E. M. Gurari, Ten-Hwang Lai, and L. E. Rosier. On Deadlock Detection in Systems of Communicating Finite State Machines. *Computers and Artificial Intelligence*, Vol.6, 1987, No.3, pp.209–228.

[5] K. Okumura. Protocol Analysis from Language Structure. In *Protocol Specification, testing, and verification VIII*, S.Aggarwal and K.Sabnani (Editors), North-Holland, 1988, pp.113–124.

[6] Jan Pachl. Protocol Description and Analysis Based on a State Transition Model with Channel Expressions, In *Protocol Specification, testing, and verification VII*, H. Rubin and C. H. West (Editors), North-Holland, 1987, pp.207–219.

[7] W. Peng and S. Purushothaman. Analysis of Communicating Processes for Non-Progress. In *Proc of the 9th IEEE International Conference on Distributed Computing Systems*, June 1989, pp.280–287.

[8] Y. T. Yu and M. Gouda. Deadlock Detection for a Class of Communicating Finite State Machines. *IEEE Transactions on Communications*, Dec 1982, pp.2514–2518.