Generalized Fair Reachability Analysis for Cyclic Protocols with Nondeterministic and Internal Transitions *

Hong Liu Raymond E. Miller

Department of Computer Science
University of Maryland at College Park
College Park, MD 20742

Abstract

In this paper, we extend the generalized fair reachability notion to cyclic protocols with nondeterministic and internal transitions. By properly incorporating internal transitions into the formulation of fair progress vectors, we prove that with only a few modifications, all the results established for cyclic protocols without nondeterministic and internal transitions still hold even if nondeterministic and internal transitions are allowed. We identify indefiniteness as a new type of logical error resulting from reachable internal execution cycles and show that it can also be detected for the class of cyclic protocols with finite fair reachable state spaces with finite extension.

1 Introduction

It is well-known that state explosion is one of the major obstacles for validating complex protocols modeled as communicating finite state machines. As a result, many techniques have been proposed to tackle this problem (please refer to [13] for a survey). It is observed that in most cases, significant state reduction can be achieved if one could eliminate as much redundancy as possible by limiting the amount of interleaving of equivalent execution sequences during state exploration. However, care must be taken to ensure that the reduced state space still maintains competitive, if not the same, logical error detecting capability as the original reachable state space.

Fair reachability analysis was originally proposed as one such improved state exploration technique for protocols with two machines [12, 4]. By forcing the two machines in a protocol to make progress at the same time, whenever possible, only fair progress states are generated during state exploration. If the fair reachable state space of a protocol is finite, detection of deadlock and unspecified reception are decidable within the fair reachable state space [12], while unboundedness detection is decidable with finite extension on the fair reachable state space [4].

In [8][1], we generalized the fair reachability notion to cyclic protocols with $n \geq 2$ machines, where each machine is deterministic but partially defined and does not have internal transitions. We showed that for a cyclic protocol P, its fair reachable state space $\mathbf{F}$ is exactly the set of reachable states with equal channel length and $\mathbf{F}$ is finite if and only if (iff for short) P is not "simultaneously unbounded". Moreover, we proved that for $\mathcal{P}$, the class of cyclic protocols whose $\mathbf{F}$'s are finite, deadlock detection is decidable within $\mathbf{F}$, while detection of other logical error, such as unspecified reception, unboundedness, and nonexecutable transition, are all decidable via finite extension of $\mathbf{F}$. We also showed that for any $P \in \mathcal{P}$, P is logically correct iff its $\mathbf{F}$ does not contain any logical errors. As a result, for class $\mathcal{P}$, our generalized fair reachability analysis technique not only can achieve substantial state reduction, but also maintains very competitive fault coverage. Therefore, it is a very useful technique for the analysis of a wide variety of cyclic protocols.

In this paper, we are going to extend our generalized fair reachability notion to the analysis of cyclic protocols where a process in a protocol can be nondeterministic but partially defined and can have internal transitions. By incorporating internal transitions into the formulation of fair progress vectors, we are able to show that with only a few modifications, all the aforementioned results established in [8] still hold for cyclic protocols with nondeterministic and internal transitions. Moreover, we observe that the inclusion of internal transitions into the model results in a new type of logical error called "indefiniteness", meaning

*Research reported in this paper was supported by NASA Grant No. NAG 5-2648.

[1] This is a revised version of the results in [6] and [7].

0-8186-7216-1/95 $04.00 © 1995 IEEE

that a protocol could reach a state from which one of the processes could indefinitely execute internal transitions without communicating with other processes in the protocol. However, we will show that indefiniteness can also be detected for the class of cyclic protocols whose **F**'s are finite, although sometimes finite extension of **F** is necessary. Therefore, our generalized fair reachability technique works equally well for the analysis of cyclic protocols with nondeterministic and internal transitions.

In [2], Cacciari and Rafiq proposed a technique called *reduced reachability analysis* that can handle internal transitions for protocols with two machines. However, unlike fair reachability analysis, two machines can proceed at the same time only if the "parallelwise" condition is satisfied. They showed that detection of deadlock and unspecified reception are decidable for a protocol with a finite reduced reachable state space. Since their approach is closely related to ours, we will defer the comparison of these two methods to Section 5, after our method is presented.

The rest of the paper is organized as follows. In the following section, the communicating finite state machine model is introduced. The fair reachability notion is generalized in Section 3, followed by the study of the logical error detection capability of fair reachable state space, We compare our work with others and discuss related issues in Section 5, and conclude the paper with ongoing and future work in Section 6.

Due to space limitations, lemmas and theorems in this paper are stated without proofs. Please refer to the full paper for detail [9].

2 The CFSM Model

Notation: (1) We use $\cdot$ to denote concatenation. Given a set M. M^* denotes its reflexive and transitive closure under concatenation. $|M|$ denotes its cardinality. 2^M denotes the *power set* of M. For $Y \in M^*$, $|Y|$ denotes its length. ϵ denotes an empty string, $|\epsilon| = 0$. (2) Given a string Y, we define $head(Y) = m$ if $Y = m \cdot Y'$; $head(Y) = \epsilon$ otherwise. (3) Given n, for any $1 \leq i \leq n$, $0 \leq j < n$, $i \oplus j = i + j$ if $i + j \leq n$ else $i \oplus j = (i + j) \bmod n$; $i \ominus j = i - j$ if $i > j$ else $i \ominus j = i - j + n$, where mod stands for the modulo operation. (4) An *interval* $[i..j]_n$ is an ordered set of at most n consecutive integers $i, i\oplus 1, \ldots, i \oplus k = j$, where $(1 \leq i \leq n) \wedge (0 \leq k < n)$. The corresponding (unordered) set is denoted as $\{i..j\}_n$. When n is known, we omit n from the notation. (5) Let $[i'..j']$ and $[i..j]$ be two intervals, $[i'..j'] \subseteq [i..j]$ iff $\{i'..j'\} \subseteq \{i..j\}$. Unless specified as $[1..n]$, we assume $|[i..j]| < n$. (6) We designate n as the number of processes in a protocol. Unless otherwise specified, we assume $n \geq 2$ and let i, j range over $\{1..n\}$.

In the communicating finite state machine (CFSM) model, a protocol is specified as a set of n processes $P = (P_1, P_2, \ldots, P_n)$, where each process P_i is a finite state machine that can communicate with other processes via FIFO channels. For each P_i, $\mathbf{S}_i$ denotes the set of local states in P_i. The *initial* local state of P_i is denoted as s_i^0. A channel from P_i to $P_j, i \neq j$, is denoted as C_{ij}. The set of messages that P_i can send to P_j is denoted as M_{ij}. The content of C_{ij}, denoted as c_{ij}, is a sequence of messages sent from P_i to P_j. When C_{ij} is empty, $c_{ij} = \epsilon$.

Let $\tilde{M}_i = (\bigcup_{j \neq i} \{-m | m \in M_{ij}\}) \cup (\bigcup_{j \neq i} \{+m | m \in M_{ji}\})$. τ denotes the partially defined *transition function*: $\bigcup_{i=1}^{n} (\mathbf{S}_i \times (\tilde{M}_i \cup \{\epsilon\}) \rightarrow 2^{\mathbf{S}_i})$, where ϵ stands for a *dummy* message. For each P_i, a transition *defined* at local state $s_i \in \mathbf{S}_i$ is denoted as $\tau(s_i, \sigma)$, where $\sigma \in \tilde{M}_i \cup \{\epsilon\}$. It is a *sending* (*receiving*) transition if $\sigma = -m$ ($\sigma = +m$); it is an *internal* transition if $\sigma = \epsilon$. As a convention, we use $\tau' = \tau(s_i, \sigma)$ to give a name τ' for this transition, and use $s_i' \in \tau(s_i, \sigma)$ to mean that s_i' is a local state resulting from the execution of the transition. We also use $\mu(s_i)$ as a shorthand for the internal transition $\tau(s_i, \epsilon)$. Denote D_i as the set of transitions defined at s_i. s_i is a *receiving* local state iff each transition in D_i is a receiving transition. By definition, each P_i is nondeterministic but partially defined.

A *transition cycle* in P_i, denoted as C_i, is a cycle in the transition graph of P_i. It is an *internal* cycle if each transition in the cycle is an internal transition. It is a *sending* (*receiving*) cycle if it is not an internal cycle and each transition in the cycle is either a sending (receiving) transition or an internal transition.

A protocol $P = (P_1, P_2, \ldots, P_n)$ is *cyclic* iff each P_i has exactly one input channel $C_{i\ominus 1,i}$ and exactly one output channel $C_{i,i\oplus 1}$. From now on, we are dealing with cyclic protocols. For results established later in this paper, it should be clear that they apply to cyclic protocols only. For ease of reference, we call a cyclic protocol P a *D-cyclic* protocol if each P_i in P is deterministic and does not contain any internal transitions.

For a cyclic protocol $P = (P_1, P_2, \ldots, P_n)$, a *global state* (*state* for short) S is represented as a $2n$-tuple $(s_1, s_2 \ldots, s_n, c_{n1}, c_{12} \ldots, c_{n-1n})$, where s_i is the local state of P_i, and $c_{i\ominus 1i}$ is the content of channel $C_{i\ominus 1i}$. In particular, the *initial state* S^0 is denoted as $(s_1^0, s_2^0 \ldots, s_n^0, \epsilon, \epsilon, \ldots, \epsilon)$. S is of *equal channel length* iff all channel contents in S are of the same length. For convenience, we use $s_i \in S$ to denote that s_i is a local state of S, and $(s_i, m) \subseteq S$ to denote that $s_i \in S$ and

$head(c_{i\ominus 1i}) = m$ in S. S is in a *sending* (*internal*) cycle iff there is a local state $s_i \in S$ that is in a sending (internal) cycle of P_i. As a convention, we use capital letters S, X to denote a state and small letter s_i to denote a local state of P_i.

The *reachability relation* among states is formulated as follows. Given two states $S = (s_1, s_2, \ldots, s_n, c_{n1}, c_{12}, \ldots, c_{n-1n})$ and $S' = (s'_1, s'_2, \ldots, s'_n, c'_{n1}, c'_{12}, \ldots, c'_{n-1n})$. S' is *directly reachable* from S, denoted as $S \mapsto S'$, iff $\exists i \in \{1..n\}$ such that the elements of S' can be derived from S by executing one of the following transitions: (1) $s'_i \in \tau(s_i, -m)$ and $c'_{ii\oplus 1} = c_{ii\oplus 1} \cdot m$. (2) $s'_i \in \tau(s_i, +m)$ and $c_{i\ominus 1i} = m \cdot c'_{i\ominus 1i}$. (3) $s'_i \in \mu(s_i)$. Except for the elements affected by the one transition applied, all other elements of S' remain the same as those in S.

Denote $\mapsto^*$ as the reflexive, transitive closure of $\mapsto$. S' is *reachable* from S iff $S \mapsto^* S'$. When $S = S^0$, we say S' is a *reachable state*. The set of reachable states in P is denoted as $\mathbf{R}$, called the *reachable state space* of P. A local state s'_i is *reachable* (from S) iff there is a state S' such that $S^0 \mapsto^* S'$ ($S \mapsto^* S'$) and $s'_i \in S'$. (s'_i, m) is *reachable* (from S) there is a state S' such that $S^0 \mapsto^* S'$ ($S \mapsto^* S'$) and $(s'_i, m) \subseteq S'$. A cycle $\mathcal{C}_i$ in P_i is *reachable* (from S) if one of the local states in $\mathcal{C}_i$ is reachable (from S).

Suppose $S^0 \mapsto^* S'$ ($S \mapsto^* S'$). An *execution sequence* of S' (from S to S'), denoted as $e = \{e_1, e_2, \ldots, e_n\}$, is a sequence $X^0 \xrightarrow{\tau^1} X^1 \xrightarrow{\tau^2} \ldots \xrightarrow{\tau^k} X^k, k \geq 0$, such that $X^0 = S^0$ ($X^0 = S$), $X^k = S'$, and $\forall l : 1 \leq l \leq k, X^{l-1} \mapsto X^l$ via transition τ^l, where each e_i is the corresponding (possibly empty) transition sequence in P_i. $\{e_1, e_2, \ldots, e_n\}$ is called a *local execution sequence set* of S' (from S to S'). The length of e, denoted as $|e|$, is defined as the number of transitions in e, i.e., $|e| = k \geq 0$. An *execution cycle* of S is a nonempty execution sequence from S to S, denoted as $\mathcal{C}_I = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$, where each $\mathcal{C}_i$ is the corresponding (possibly empty, not necessarily elementary) transition cycle in P_i, and I is the index set for those nonempty $\mathcal{C}_i$'s in $\mathcal{C}$. By definition, $|I| > 0$. $\mathcal{C}_I$ is an *internal* execution cycle of S iff $\forall i \in I : \mathcal{C}_i$ is an internal cycle in P_i.

For protocol validation, we check $\mathbf{R}$ against common errors defined below. Given a state $S \in \mathbf{R}$, let $S = (s_1, s_2 \ldots, s_n, c_{n1}, c_{12} \ldots, c_{n-1n})$: S is a *deadlock* state iff each $s_i \in S$ is a receiving local state in P_i and each channel $C_{i\ominus 1i}$ is empty; S is an *unspecified reception* state iff there is an $s_i \in S$ such that s_i is a receiving local state, $head(c_{i\ominus 1i}) = m$, and $\tau(s_i, +m) \notin D_i$; S is an *indefinite* state iff S is in an internal cycle. P is *unbounded* iff $\forall K \geq 0 : \exists S \in \mathbf{R} : \exists i \in \{1..n\} : |c_{i\ominus 1i}| > K$. A transition $\tau(s_i, \sigma) \in D_i$ is *executable* in S iff $\sigma = -m$, or $\sigma = \mu$, or $\sigma = +m$ and $head(c_{i\ominus 1i}) = m$. $\tau(s_i, \sigma)$ is *nonexecutable* iff it is not executable in each $S \in \mathbf{R}$. Deadlock, unspecified reception, nonexecutable transition, unboundedness, and indefiniteness are called *logical errors*. P is *logically correct* iff $\mathbf{R}$ is free of logical errors.

It should be clear that the inclusion of internal transitions into the model introduces indefiniteness as a new type of logical error, meaning that from an indefinitely state, a process could loop indefinitely through its internal cycle without communicating with its neighbors. It can be shown that none of the logical errors is decidable for cyclic protocols in general, using the results established in [1].

3 Fair Reachability Analysis

Fair reachability was generalized to D-cyclic protocols with $n \geq 2$ machines in [8]. In this section, we will show how the fair reachability notion for D-cyclic protocols can be extended to cope with nondeterministic and internal transitions for general cyclic protocols. For the sake of space, we will be expanding on the modification part of the formulation and be brief on the part that is unchanged. Please refer to [8] for a complete treatment. For conciseness, we use "fair reachability" for "generalized fair reachability" from now on.

Given a cyclic protocol $P = (P_1, P_2, \ldots, P_n)$. Let $S = (s_1, s_2 \ldots, s_n, c_{n1}, c_{12} \ldots, c_{n-1n})$ be a state of P. Define $E_i^- = \{\tau(s_i, -m) \in D_i\}$, $E_i^+ = \{\tau(s_i, +m) \in D_i | head(c_{i\ominus 1i}) = m\}$, $E_i^\mu = \{\mu(s_i \in D_i)\}$, and $E_i = E_i^- \cup E_i^+ \cup E_i^\mu$. Then E_i is the set of executable transitions at s_i in S. A transition $\tau(s_i, +m) \in D_i$ is *enabled* in S iff $(c_{i\ominus 1i} = \epsilon) \wedge (\tau(s_{i\ominus 1}, -m) \in D_{i\ominus 1})$. Let E_i^{++} be the set of enabled transitions at s_i in S.

Convention: The notations defined above are implicitly bound to a state S. For brevity, S is dropped from the notations when S is given and no confusion arises. This convention is adopted throughout the paper when a new notation is introduced. Whenever distinction is necessary, the binding arguments, such as S, will be put into the notation. For example, when we talk about the set of executable transitions in P_i in both S^1 and S^2, we will use $E_i(S^1)$ and $E_i(S^2)$, respectively.

Let T be a set of executable transitions in S such that $\forall t, t' \in T : (t \in E_i) \wedge (t' \in E_i) \Rightarrow t = t'$. A *linear ordering* of T is an execution sequence e where each transition of T occurs exactly once in e. The following lemmas are obvious.

Lemma 3.1 Given S and T, let e and e' be two linear orderings of T. S' is reachable from S via e iff S' is reachable from S via e'.

Thus, we say that S' is reachable from S via T iff S' is reachable from S via an linear ordering of T.

Lemma 3.2 Given S and T, suppose $S \mapsto^* S'$. For each execution sequence e from S to S'. Let $|e| = k$ and $\tau^1 \cdot \tau^2 \cdots \tau^k$ be the corresponding transition sequence in e. Assume that $\tau^j \in T$ and $j \neq 1$. Then S' is also reachable from S via another execution sequence e' whose corresponding transition sequence is $\tau^j \cdot \tau^1 \cdots \tau^{j\ominus 1} \cdot \tau^{j\oplus 1} \cdots \tau^k$ when $j \neq k$; $\tau^j \cdot \tau^1 \cdots \tau^{k\ominus 1}$ otherwise.

Lemma 3.3 Given S, S', and T. Suppose $S \mapsto^* S'$ and $T = \{\tau^1, \tau^2, \ldots, \tau^j\}$. If each transition in T appears in an execution sequence e from S to S', then there is an execution sequence e' from S to S' whose corresponding transition sequence is $\tau^1 \cdots \tau^j \cdot \tau^{j\oplus 1} \cdots \tau^k$, where $|e| = k$ and $\tau^{j\oplus 1}, \ldots, \tau^k$ are the remaining transitions in e.

Lemma 3.4 Given S, S', and T. Suppose $S \mapsto^* S'$ via e, each transition in T occurs in e, and $S \mapsto^* S''$ via an linear ordering of T. Then $S'' \mapsto^* S'$ via the remaining transitions in e.

The basic idea of fair reachability analysis is to let more than one processes progress at the same time in S, whenever possible. However, in order to maintain a certain channel property, we should be careful in selecting T for simultaneous transition executions in S. In what follows, we are going to describe a selection scheme with which the equal channel length property is preserved.

Given a state $S = (s_1, s_2, \ldots, s_n, c_{n1}, c_{12}, \ldots, c_{n-1n})$. Denote λ as a *null* transition, indicating no state change in a process. Define $T_i = E_i$ if $(E_i \neq \emptyset) \wedge (c_{i\ominus 1 i} \neq \epsilon)$; $T_i = \{\lambda\}$ if $(E_i = \emptyset) \wedge (c_{i\ominus 1 i} = \epsilon)$; $T_i = D_i$ otherwise. Let $TV = \{\vec{\mathbf{t}} = (t_1, t_2, \ldots, t_n)\} \subseteq \times_{k=1}^{n} T_k$ such that (i) $\exists t_i \in E_i$, and (ii) $\forall i \in \{1..n\}$, if $(t_i \in E_i^-) \wedge (t_{i\oplus 1} \in E_{i\oplus 1}^{++})$, then $(t_i = (s_i, -m)) \wedge (t_{i\oplus 1} = \tau(s_{i\oplus 1}, +m))$. For each $\vec{\mathbf{t}} \in TV$, we compute another transition vector $\vec{\mathbf{v}} = (v_1, v_2, \ldots, v_n)$ according to one of the following four cases:

(1) $\vec{\mathbf{t}} \in (\times_{k=1}^{n} E_k^-) \cup (\times_{k=1}^{n} E_k^+)$. In this case, $\vec{\mathbf{v}} = \vec{\mathbf{t}}$. Here, $\vec{\mathbf{v}}$ is called a *concurrency vector* in S.

(2) $\exists j : (t_j \in E_j^-) \wedge (t_{j\oplus 1} \in E_{j\oplus 1}^+ \cup E_{j\oplus 1}^{++})$. $(t_{j\ominus 1}, t_j)$ is called a *send-receive pair* in $\vec{\mathbf{t}}$. In this case, $v_i = t_i$ if t_i is in a send-receive pair or $t_i = \mu(s_i)$; $v_i = \lambda$ otherwise. $\vec{\mathbf{v}}$ is called a *synchronization* vector in S.

(3) $\exists j : t_j = \mu(s_j)$ and there is no send-receive pair in $\vec{\mathbf{t}}$. In this case, $v_i = t_i$ if $t_i = \mu(s_i)$; $v_i = \lambda$ otherwise. $\vec{\mathbf{v}}$ is called a *internal* vector in S.

(4) None of conditions (1) through (3) hold. In this case, set $v_i = \lambda$. The resulting vector is called a *null* vector, denoted as $\vec{\lambda}$.

$\vec{\mathbf{v}}$ thus computed is called a *fair progress* vector in S if it is not a null vector. The set of concurrency (synchronization, internal) vectors in S is denoted as V_c (V_s, V_μ). Let $V = V_c \cup V_s \cup V_\mu$.

Given S and $\vec{\mathbf{v}} \in V$, the *execution* of $\vec{\mathbf{v}}$ in S is defined as follows: First, all the executable transitions in $\vec{\mathbf{v}}$ are executed in any linear order to get to a state S'' from S. Next, all the remaining transitions, if any, that are enabled in S (and thus executable in S'') are executed in any linear order to get to S' from S''. By Lemma 3.1, S'' and S' are unique with respect to (w.r.t for short) S and $\vec{\mathbf{v}}$. As a result, the execution of $\vec{\mathbf{v}}$ in S is well-defined. In this case, we say S' is *directly fair reachable* from S, denoted as $S \mapsto_f S'$. Let $\mapsto_f^*$ be the reflexive, transitive closure of $\mapsto_f$. S' is *fair reachable* from S iff $S \mapsto_f^* S'$. When $S = S^0$, S' is fair reachable. We can also define fair reachability for s_i, (s_i, m), $\mathcal{C}_i$, and $\mathcal{C}_I$ in much the same way as their reachability counterparts.

Note that $S^0 \in \mathbf{F}$ is a state with equal channel length of zero. It is not difficult to show that any fair progress vector in S^0 maintains the equal channel length property in the resulting state. Inductively, we have the following result:

Theorem 3.1 Each fair reachable state is a reachable state with equal channel length.

Note that the converse of this theorem is not true. This indicates that our fair reachability technique can offer substantial state reduction during state exploration. On the other hand, for each reachable state S, we can always find a fair reachable state that is closest to S w.r.t a local execution sequence set $\{e_1, e_2, \ldots, e_n\}$ of S. Specifically, we construct a *partial fair execution sequence* for S w.r.t $\{e_1, e_2, \ldots, e_n\}$, denoted as $pfs = X^0 \xrightarrow{\vec{\mathbf{v}}_1} X^1 \xrightarrow{\vec{\mathbf{v}}_2} \cdots \xrightarrow{\vec{\mathbf{v}}_k} X^k$, such that $k \geq 0$, $X^0 = S^0$, $\forall\ 0 < l \leq k : X^{l-1} \mapsto_f X^l$ via fair progress vector $\vec{\mathbf{v}}_l$, and no fair progress vector can be derived from $\{e_1, e_2, \ldots, e_n\}$ in state X^k. X^k is called the *fair precursor* of S w.r.t $\{e_1, e_2, \ldots, e_n\}$, denoted as $fp = (s_1^p, s_2^p \ldots, s_n^p, c_{n1}^p, c_{12}^p \ldots, c_{n-1n}^p)$. It is not difficult to show that pfs and fp are *unique* w.r.t $\{e_1, e_2, \ldots, e_n\}$.

Lemma 3.5 Let fp be the fair precursor for a reachable state S w.r.t $\{e_1, e_2, \ldots, e_n\}$. If $fp \neq S$,

then the following statements are true in fp: (1) $\exists k \in [1..n] : |e_k| \neq 0$. (2) $\exists k \in [1..n] : |e_k| = 0$. (3) If $|e_k| \neq 0$, let τ_k^p be the transition from e_k at s_k^p, then τ_k^p is executable. (4) $fp \mapsto^* S$ via the remaining transitions from $\{e_1, e_2, \ldots, e_n\}$ in fp.

We are primarily interested in the class of cyclic protocols whose fair reachable state spaces are finite. In [8], we showed that for a D-cyclic protocol, **F** is finite iff it is not "simultaneously unbounded". (A cyclic protocol P is *simultaneously unbounded* iff $\forall K \geq 0 \exists K' > K$ such that there is a state $S \in \mathbf{R}$ where each channel has length no less than K'.) By similar arguments, we can also show that the same necessary and sufficient condition holds for cyclic protocols in general, although it is also undecidable.

Lemma 3.6 Given a cyclic protocol P without reachable sending cycles. If P is unbounded, then P is simultaneously unbounded.

Lemma 3.7 If a cyclic protocol P is simultaneously unbounded, then its **F** is infinite.

Theorem 3.2 Given a cyclic protocol P with a finite **F**. P is unbounded iff it has a reachable sending cycle.

Theorem 3.3 Given a cyclic protocol P. **F** is finite iff P is not simultaneously unbounded.

Theorem 3.4 It is undecidable whether a cyclic protocol P has a finite **F**.

By Theorem 3.2, we define a reachable state S as an *unbounded* state iff it is in a sending cycle. In [8], we use $\mathcal{P}$ to denote the class of D-cyclic protocols whose **F**'s are finite. In this paper, we use $\mathcal{Q}$ to denote the class of cyclic protocols whose **F**'s are finite. From the preceding discussion, we know that $\mathcal{Q}$ maintains the same equal channel length property and membership function as $\mathcal{P}$. From now on, we will restrict our study to class $\mathcal{Q}$. In the rest of the paper, unless otherwise stated explicitly, when we mention a cyclic protocol P, we mean $P \in \mathcal{Q}$; when we mention **F**, we mean that it is finite.

4 Fault Coverage of F

As we have already seen, **F** is usually much smaller than **R**. For fair reachability to be useful, we need to make sure that the reduced state space **F** remains competitive in terms of fault coverage for the class of cyclic protocols $\mathcal{Q}$. By taking indefiniteness into account, we are going to show that those results established for $\mathcal{P}$ in [8] are still valid for $\mathcal{Q}$, except for a few minor changes. Since the line of reasoning is similar to that used in [8], we will be quite informal in the arguments we make and highlight the differences along the way. Interested readers should refer to [8] for detail.

Let's first study deadlock detection. Notice that each deadlock state is of equal channel length zero. Although not all the reachable states with equal channel length zero are fair reachable, a deadlock state dose not have any executable transitions. This special property enable us to show that each deadlock state is fair reachable, based on Lemma 3.5.

Theorem 4.1 Deadlock detection is decidable for $\mathcal{Q}$.

By Theorem 3.2, it is sufficient to detect reachable sending cycles for unboundedness detection for $\mathcal{Q}$. Similarly, to detect indefiniteness, it is sufficient to detect reachable internal cycles. Both problems can be reduced to the following local state reachability problem [4, 8]:

P-I : *Given a local state s_j, decide whether s_j is reachable.*

Note that in some cases, fair progress can detect unspecified reception in a "look-ahead" manner. Specifically, we regard a fair reachable state $S = (s_1, s_2, \ldots, s_n, c_{n1}, c_{12}, \ldots, c_{n-1n})$ as a (fair) unspecified reception state if there exists an $s_j \in S$ such that s_j is a receiving local state, $c_{j \ominus 1 j} = \epsilon$, $\tau(s_{j \ominus 1}, -m) \in D_{j \ominus 1}$, and $\tau(s_j, +m) \notin D_j$. By definition, detection of unspecified reception can be reduced to the following local state reachability problem:

P-II *Given a local state s_j and a message $m \in M_{j \ominus 1 j}$, decide whether (s_j, m) is reachable.*

As for nonexecutable transition detection, for a sending transition $\tau(s_j, -m) \in D_j$, the problem can be reduced to **P-I**; for a receiving transition $\tau(s_j, +m) \in D_j$, the problem can be reduced to **P-II**. As a result, detection of all the logical errors other than deadlock can be reduced to **P-I** and/or **P-II**.

Note that it is possible to have a fair reachable state S whose $V = \emptyset$. In this case, it can be shown that S is either a deadlock state or a (fair) unspecified reception state, and thus will not introduce any new type of logical error in **F**.

Unfortunately, not each reachable local state s_j is fair reachable. Thus, to show the decidability of **P-I** for $\mathcal{Q}$, **F** needs to be finitely extended to uncover

those local states. The same argument can be made for **P-II**. In what follows, we are going to show that both **P-I** and **P-II** are decidable for $\mathcal{Q}$. For the sake of space, we will focus on **P-I**. The proof for **P-II** can be formulated in a similar way.

Suppose s_j is reachable but not fair reachable. Then none of the reachable states containing s_j is in $\mathbf{F}$. Let S be any reachable state with $s_j \in S$, and $\{e_1, e_2, \ldots, e_n\}$ be a local execution sequence set for S. Let pfs and fp be the partial fair execution sequence and the fair precursor for S w.r.t $\{e_1, e_2, \ldots, e_n\}$, respectively. Denote $fp = (s_1^p, s_2^p \ldots, s_n^p, c_{n1}^p, c_{12}^p \ldots, c_{n-1n}^p)$. By Lemma 3.5, we can find a maximal interval $[i..j]$ in fp such that $\forall k \in [i..j] : |e_k| \neq 0$ and $|e_{i\ominus 1}| = 0$ in fp. Moreover, let τ_k^p be the transition from e_k at s_k^p, then τ_k^p is executable in fp.

Starting from fp, we construct the set of states fair reachable from fp as follows: In each such state S', each fair progress vector $\vec{\mathbf{v}}$ is computed as usual except that v_k must take on the transition from e_k if $(k \in [i..j]) \wedge (|e_k| \neq 0)$ in that state. Note that some of the e_k's might become empty during the process, but by no means can e_j become empty. Without loss of generality, let's assume that none of the e_k's becomes empty during the construction. Let $\mathbf{F}_{[i..j]}^{min}$ be the set of states from the construction where sum of the remaining transitions in $\{e_i, e_{i\oplus 1}, \ldots, e_j\}$ is minimum. Note that if $S' \in \mathbf{F}_{[i..j]}^{min}$ and S'' is fair reachable from S' by the construction, then $S'' \in \mathbf{F}_{[i..j]}^{min}$. More importantly, S'' is fair reachable from S' *without progress* in $[i..j]$.

Lemma 4.1 Given a local state s_j ((s_j, m)). s_j ((s_j, m)) is reachable but not fair reachable only if there is a fair reachable state S' such that S' does not have a concurrency vector and there is a transition vector $\vec{\mathbf{u}}_{[i..j]} = (u_i, u_{i\oplus 1}, \ldots, u_j)$ in S' satisfying the following five conditions: (i) $\forall k \in [i..j] : u_k \in E_k$; (ii) there is no send-receive pair in $\vec{\mathbf{u}}_{[i..j]}$; (iii) neither u_i nor u_j appears in a send-receive pair in any synchronization vector in S'; (iv) s_j is reachable from S' via a local execution sequence set $\{e_1', e_2', \ldots, e_n'\}$ such that $\forall k \in [i..j] : u_k$ is the first transition in e_k'; and (v) for each S'' fair reachable from S' without progress in $[i..j]$, $\vec{\mathbf{u}}_{[i..j]}$ is a transition vector in S'' satisfying (i)–(iv).

A transition vector satisfying conditions (i)–(v) in the above lemma is called a *persistent incompatible transition vector* (*pitv* for short) in S'. Therefore, the extension of $\mathbf{F}$ should be based on the set of states in $\mathbf{F}$ each of which has a *pitv* $\vec{\mathbf{u}}_{[i..j]}$ for some interval $[i..j]$. This set is called a *extension set* of $\mathbf{F}$, denoted as $\mathbf{F}_T$. Thanks to the following lemma, $\mathbf{F}_T$ can be easily identified and its size can be greatly reduced. Please refer to [7, 8] for detail.

Lemma 4.2 Given $S \in \mathbf{F}$ and a transition vector $\vec{\mathbf{u}}_{[i..j]}$. $\vec{\mathbf{u}}_{[i..j]}$ is a *pitv* in S only if there is another $S' \in \mathbf{F}$ such that $\vec{\mathbf{u}}_{[i..j]}$ is a *pitv* in S' and S' is either a (fair) unspecified reception state, or an unbounded state, or an indefinite state.

Denote $\mathbf{F}_{dl}$ ($\mathbf{F}_{ur}$, $\mathbf{F}_{ub}$, $\mathbf{F}_{id}$) as the set of deadlock (unspecified reception, unbounded, indefinite) states in $\mathbf{F}$. By the above lemma, $\mathbf{F}_T = \mathbf{F}_{ur} \cup \mathbf{F}_{ub} \cup \mathbf{F}_{id}$. Clearly, $\mathbf{F}_T$ can be easily computed during the construction of $\mathbf{F}$. From Lemma 4.1 and 4.2, $\mathbf{F}_T$ is exactly the extension set we want. Thus to solve both **P-I** and **P-II** for $\mathcal{Q}$, we only need to finitely extend those states in $\mathbf{F}_T$, which can be done in a similar way as for D-cyclic protocols. Again, please refer to [7, 8] for detail.

Theorem 4.2 Both **P-I** and **P-II** are decidable for $\mathcal{Q}$. Therefore, detection of unspecified reception, unboundedness, nonexecutable transition, and indefiniteness are all decidable for $\mathcal{Q}$.

During the process, we have also found out a fault coverage characterization for $\mathbf{F}$ similar to that for $\mathbf{F}$ of a D-cyclic protocol [8]. The only difference is that we need to take indefiniteness into account.

Theorem 4.3 Given a cyclic protocol $P \in \mathcal{Q}$. P has a deadlock iff $\mathbf{F}_{dl} \neq \emptyset$. P has an unspecified reception but $\mathbf{F}_{ur} = \emptyset$ only if $\mathbf{F}_{ub} \cup \mathbf{F}_{id} \neq \emptyset$. P is unbounded but $\mathbf{F}_{ub} = \emptyset$ only if $\mathbf{F}_{ur} \cup \mathbf{F}_{id} \neq \emptyset$. P is indefinite but $\mathbf{F}_{id} = \emptyset$ only if $\mathbf{F}_{ur} \cup \mathbf{F}_{ub} \neq \emptyset$. P has a nonexecutable transition that is not detectable via $\mathbf{F}$ only if $\mathbf{F}_{ur} \cup \mathbf{F}_{ub} \cup \mathbf{F}_{id} \neq \emptyset$. P is logically correct iff $\mathbf{F}$ does not contain any logical errors.

As a result, $\mathbf{F}$ not only can offer substantial state reduction over $\mathbf{R}$ but also is very competitive in fault coverage. Theorem 4.3 also suggests the following iterative validation strategy: We first generate $\mathbf{F}$. If $\mathbf{F}$ is error-free, then we are done; otherwise we fix the errors detected in $\mathbf{F}$ and regenerate $\mathbf{F}$. This process is repeated until no errors are caught in $\mathbf{F}$. At the end of the process, we will have a cyclic protocol that is logically correct. By doing so, finite extension of $\mathbf{F}$ is in fact not needed at all.

5 Discussion

While the notion of internal transitions has been used extensively in other models such as the *labeled*

transition systems (LTS for short) model, the study of this notion in the CFSM model has been limited. The reduced reachability analysis approach by Cacciari and Rafiq [2] seems to be most closely related to the present work in that reduced progress is quite similar to fair progress and internal transitions are allowed in the model. However, there are several differences between their approach and ours: First, reduced reachability analysis was proposed for protocols with $n = 2$ machines. It remains to be shown whether this technique can be generalized to protocols with $n \geq 2$ machines. In addition, they assumed no internal cycles in the protocols. Although it seems that their technique can also be extended to handle internal cycles, it is not clear why such a restriction was made in their formulation. Second, the "parallelwise" condition imposed on reduced reachability implies that it is not always possible for the two machines to proceed simultaneously. Thus not every reduced reachable state is of equal channel length. This, we feel, makes it more difficult to find a (sufficient) condition for the class of protocols with finite reduced reachable state spaces. Third, since each channel is empty in the initial state and only one machine is allowed to proceed when both channels are empty in reduced state exploration, the reduced reachable state space properly includes the fair reachable state space if the protocol has more than one reachable state, as is the case for most protocols. For $n = 2$, although their approach can detect deadlocks and unspecified receptions without extending the reduced reachable state space, fair reachability analysis can accomplish the same task without finite extension and generates fewer states for most protocols. Even if their technique can be generalized to cyclic protocols with $n \geq 2$ machines and can still detect unspecified receptions within the reduced reachable state space, it is not clear whether the saving in finite extension can be paid off by generating more states. Besides, it remains to be seen whether detection of unboundedness and nonexecutable transitions can be done using their approach. To sum up, compared with reduced reachability analysis, our approach has the advantage that it can be applied to a much larger class of protocols, can detect more types of logical errors in a protocol, and is quite efficient in terms of both space and time.

Based on the preceding presentation, it is clear that fair reachability analysis is an effective validation technique for cyclic protocols with nondeterministic and internal transitions. This study, together with the one on D-cyclic protocols [8], revealed the following important relationship between $\mathbf{F}$ and $\mathbf{R}$ (see Lemma 3.5), which lays the foundation for our study on cyclic protocol validation:

$\forall S \in \mathbf{R} : \exists S' \in \mathbf{F} : \exists i \in \{1..n\} : S' \mapsto^* S$ *without progress in* P_i.

In [10], we study the following three general reachability problems for cyclic protocols:

P-S: *Given a (global) state S, decide whether S is reachable.*

P-A: *Given an abstract state $A = (s_1, s_2, \ldots, s_n, m_1, m_2, \ldots, m_n)$, where $m_i \in M_{i\ominus 1 i} \cup \{\epsilon\}$, decide whether A is reachable, i.e., whether there is a reachable state $S' = (s'_1, s'_2, \ldots, s'_n, c'_{n1}, c'_{12}, \ldots, c'_{n-1n})$ such that $\forall i \in \{1..n\} : (s_i = s'_i) \wedge (m_i = head(c'_{i\ominus 1 i}))$.*

P-C: *Given an execution cycle $\mathcal{C}_I$, decide whether there is a reachable state S such that $\mathcal{C}_I$ is an execution cycle of S.*

These three problems are called *the global state reachability problem, abstract state reachability problem, and execution cycle reachability problem*, respectively. **P-S** by itself is of general interest. **P-A** is a generalization of both **P-I** and **P-II**, and **P-C** is a generalization of livelock detection [3, 6]. Using maximal progress state exploration [5, 8], we showed that for a given $S' \in \mathbf{F}$ and an $i \in \{1..n\}$, the following three problems are decidable for cyclic protocols: (i) Whether a given S is reachable from S' without progress in P_i; (ii) Whether a given A is reachable from S' without progress in P_i; (iii) Whether a given $\mathcal{C}_I$ is reachable from S' without progress in P_i. Based on the relationship between $\mathbf{F}$ and $\mathbf{R}$, it follows that **P-S**, **P-A**, and **P-C** are all decidable for $\mathcal{Q}$. Please refer to [10] for detail.

Assume $k \in \{1..n\}$. A protocol $P = (P_1, P_2, \ldots, P_n)$ has a *k-livelock* iff there is a reachable execution cycle $\mathcal{C}_I$ such that $|I| = k$ and $\forall i \in I : \mathcal{C}_i$ is marked nonprogress in P_i. P is *k-indefinite* iff there is a reachable execution cycle $\mathcal{C}_I$ such that $|I| = k$ and $\forall i \in I : \mathcal{C}_i$ is an internal cycle in P_i. As a corollary of decidability of **P-C**, both k-livelock and k-indefiniteness are decidable for $\mathcal{Q}$.

6 Conclusion

In this paper, we extended the generalized fair reachability analysis technique to cyclic protocols with nondeterministic and internal transitions. We showed that except for a few minor changes, all the results established for cyclic protocols in [8] can be carried over to cyclic protocols with nondeterministic and internal transitions. We identified indefiniteness as a

new type of logical error and showed that its detection is also decidable for $\mathcal{Q}$ via finite extension of the fair reachable state space. As a result, our technique works equally well for the class of cyclic protocols with finite fair reachable state spaces even if nondeterministic and internal transitions are allowed.

However, the cyclic protocols are still very restricted in terms of communication topology. It will be nice if we can generalize the fair reachability technique to protocols with more complicated, and yet regular communication topologies. As the first step, our ongoing research involves extending the fair reachability notion to a class of *multi-cyclic* protocols, where each protocol is composed of a set of component cyclic protocols interconnected in such a way that each channel belongs to exactly one component cyclic protocol, but a process might belong to a number of component cyclic protocols [11]. The basic idea is to first compute the set of fair progress vectors for each component cyclic protocol, then composed them together to form the set of (global) fair progress vectors. Thanks to the interconnection constraints, the composition process can be shown to be feasible. Based on this formulation, we showed in [11] that each fair reachable state in a multi-cyclic protocol is of equal channel length w.r.t each component cyclic protocol (but two component cyclic protocols might be of different channel length in the same state), and each deadlock state is fair reachable. Therefore, deadlock detection is decidable for the class of multi-cyclic protocols whose fair reachable state spaces are finite. We are currently studying the fault coverage capability of the fair reachable state space of a multi-cyclic protocol for logical errors other than deadlock.

We are also investigating the possibility of applying the fair reachability technique to other formal models such as the extended finite state machines model.

Acknowledgement

The authors would like to thank the anonymous referees for their comments that help improve the presentation of this paper.

References

[1] D. Brand and P. Zafiropulo, "On Communicating Finite-State Machines," Journal of ACM, Vol. 30, No. 2, April 1983, pp. 323–342.

[2] L. Cacciari and O. Rafiq, "On Improving Reduced Reachability Analysis," FORTE'92, Perros-Guirec, France, October 13-16, 1992, M. Daiz and R. Groz (Ed.), 1992, pp. 137–152.

[3] M.G. Gouda, C.H. Chow, and S.S. Lam, "Livelock Detection in Networks of Communicating Finite State Machines," Technical Report, TR-84-10, Dept. of Computer Science, Univ. of Texas at Austin, April 1984.

[4] M.G. Gouda and J.Y. Han, "Protocol Validation by Fair Progress State Exploration," Computer Networks and ISDN Systems, Vol. 9, 1985, pp. 353–361.

[5] M.G. Gouda and Y.T. Yu, "Protocol Validation by Maximal Progress State Exploration," IEEE Transactions on Communications, Vol. COM-32, No. 1, 1984, pp. 94–97.

[6] H. Liu and R.E. Miller, "Generalized Fair Reachability Analysis for Cyclic Protocols: Part 1," PSTV'94, S.T. Vuong (Ed.), Vancouvor, B.C. Canada, June 1994, pp. 258–273.

[7] H. Liu and R.E. Miller, "Generalized Fair Reachability Analysis for Cyclic Protocols: Decidability for Logical Correctness Problems," ICNP'94, Boston, Massachusetts, October 25–28, pp. 100–107.

[8] H. Liu and R.E. Miller, "Generalized Fair Reachability Analysis for Cyclic Protocols," *submitted for journal publication.*

[9] H. Liu and R.E. Miller, "Generalized Fair Reachability Analysis for Cyclic Protocols with Nondeterministic and Internal Transitions," *in preparation.*

[10] H. Liu and R.E. Miller, "Reachability Problems for Cyclic Protocols," Accepted for 1995 International Conference on Communications and Computer Networks (IC3N'95).

[11] H. Liu and R.E. Miller, "Deadlock Detection for Multi-Cyclic Protocols via Generalized Fair Reachability Analysis," *in preparation.*

[12] J. Rubin and C.H. West, "An Improved Protocol Validation Technique," Computer Networks and ISDN Systems, Vol. 6, 1982, pp. 65–73.

[13] D. Sidhu, A. Chung, and T.P. Blumer, "Experience with Formal Methods in Protocol Development," ACM SIGCOMM, Computer Communication Review, Vol. 21, No. 2, April, 1991, pp. 81–101.