Generalized Fair Reachability Analysis for Cyclic Protocols with Nondeterminism 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 nondeterminism and internal transitions. By properly incorporating internal transitions into the formulation of fair progress vectors, we prove that most of the results established for cyclic protocols without nondeterminism and internal transitions still hold even if nondeterminism and internal transitions are allowed. We identify *indefiniteness* as a new type of logical error resulting from reachable internal execution cycles and show that indefiniteness can also be detected for the class of cyclic protocols with finite fair reachable state spaces with finite extensions.

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 [10] 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 [9, 6]. 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

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

deadlock and unspecified reception are decidable within the fair reachable state space [9], while unboundedness detection is decidable with finite extension on the fair reachable state space [6].

In [7, 8], 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 $\mathbf{F}$ [7], while detection of other logical error, such as unspecified reception, unboundedness, and nonexecutable transition, are all decidable via finite extension of $\mathbf{F}$ [8]. We also showed that for any $P \in \mathcal{P}$, P is logically correct iff its $\mathbf{F}$ does not contain any logical errors [8]. 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 all the aforementioned results for cyclic protocols without nondeterminism and internal transitions still hold for cyclic protocols with nondeterminism 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 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 $\mathbf{F}$'s are finite, although sometimes finite extension of $\mathbf{F}$ is necessary. Therefore, our generalized fair reachability technique works equally well for the analysis of cyclic protocols with nondeterminism and internal transitions.

In [2, 3], 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. In Section 3, we extend the generalized fair reachability notion to cyclic protocols with nondeterminism and internal transitions and study the basic properties of fair reachable state space. We study the logical error detection capability of fair reachable state space in Section 4, where we show how finite extension can be performed on a finite fair reachable space so that logical errors other than deadlock, including indefiniteness, can be detected effectively and efficiently. We conclude the paper with 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 details.

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 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. (3) An *interval* $[i..j]$ 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 $\{i..j\}$. 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 $|[i..j]| < n$. (4) 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*: $\bigcup_{i=1}^{n} (\mathbf{S}_i \times \tilde{M}_i \rightarrow 2^{\mathbf{S}_i})$. For each P_i, a transition *defined* at local state $s_i \in \mathbf{S}_i$ is denoted as $\tau(s_i, \sigma)$, where $\sigma \in \tilde{M}_i$. It is a *sending* (*receiving*) transition if $\sigma = -m$ ($\sigma = +m$). 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. Similarly, μ denotes the partially defined *internal transition function*: $\bigcup_{i=1}^{n} (\mathbf{S}_i \rightarrow 2^{\mathbf{S}_i})$. For each P_i, an internal transition defined at s_i is denoted as $\mu(s_i)$. We also use $s_i' \in \mu(s_i)$ to mean that s_i' is a local state resulting from the execution of μ at s_i. By definition, each P_i is nondeterministic but partially defined.

A *transition cycle* in P_i, denoted as $\mathcal{C}_i$, 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 in P_i if it is not an internal cycle and each transition in the cycle is either a sending (receiving) transition or an internal transition. s_i is a *sending* (*receiving*) local state iff all transitions defined in s_i are sending (receiving) transitions.

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 $C_{ii\oplus 1}$. 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 $c_{i\ominus 1 i}$ is the content of channel $C_{i\ominus 1 i}$. 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 $(m, s_i) \in S$ to denote that $s_i \in S$ and m is at the head of channel $C_{i\ominus 1 i}$ in S. S is in a *sending* cycle iff there is a local state $s_i \in S$ that is in a sending 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 $c_{i\ominus 1 i} = m \cdot c'_{i\ominus 1 i}$. (3) $s'_i \in \tau(s_i)$. 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 $s'_i \in S'$. (m, s'_i) is *reachable* (from S) there is a state S' such that $S^0 \mapsto^* S'$ ($S \mapsto^* S'$) and $(m, s'_i) \in S'$. 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 $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$, where each $\mathcal{C}_i$ is the corresponding (possibly empty, not necessarily elementary) transition cycle in P_i. $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ is called a *local execution cycle set* of S. By definition, at least one $\mathcal{C}_i$ is not empty. $\mathcal{C}$ is an *internal* execution cycle of S iff each nonempty $\mathcal{C}_i$ is an internal cycle in P_i.

For protocol validation, we check $\mathbf{R}$ against common errors such as deadlock, unspecified reception, nonexecutable transition, and unboundedness. (For definitions of these concepts, please refer to [1].) The inclusion of internal transitions in the model introduces a new type of logical error resulting from *internal execution cycles*. By definition, P has an internal execution cycle iff one of the processes in P has a reachable internal cycle. A reachable state S is an *indefinite* state iff there is an $s_i \in S$ that is in an internal cycle of P_i, meaning that from S and on, there is a process that could loop indefinitely through its internal cycle without communications with its neighbors. A protocol P is *indefinite* iff it has an indefinite state. It should be clear that indefiniteness is also a syntactic error that can be checked in the same way as other aforementioned errors by inspecting each reachable state during state exploration. 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 can be shown that none of the logical errors is decidable for cyclic protocols in general using the results established in [1].

3 Generalized Fair Reachability Analysis

Fair reachability was generalized to D-cyclic protocols with $n \geq 2$ machines in [7, 8]. In this section, we first show how the fair reachability notion for D-cyclic protocols can be extended to cope with nondeterminism and internal transitions for general cyclic protocols. Then we show that the fair reachable state space still maintains the equal channel length property and satisfies the same necessary and sufficient condition for being finite. 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 [7, 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. Denote E_i^- as the set of sending transitions defined at s_i. Define $E_i^+ = \{\tau(s_i, +m)\}$ if $(m, s_i) \in S$; $E_i^+ = \emptyset$ otherwise. Let $E_i = E_i^- \cup E_i^+$. Then E_i is the set of executable transitions at s_i in S. We also define E_i^{++} as the set of enabled transitions at s_i in S. ($\tau(s_i, \sigma)$ is *enabled* iff $\sigma = +m, c_{i\ominus 1i} = \epsilon$, and $\tau(s_{i\ominus 1}, -m)$ is defined.)

Denote λ as the *null* transition, indicating no state change in a process. Let $TV = \{\vec{\mathbf{t}} = (t_1, t_2, \ldots, t_n)\}$ such that (i) $\forall\, i \in [1..n] : t_i \in E_i \cup E_i^{++}$ if $E_i \cup E_i^{++} \neq \emptyset$; $t_i = \lambda$ otherwise, and (ii) $\forall\, i \in [1..n]$, if $t_i = (s_i, +m) \in E_i^{++}$, then $t_{i\ominus 1} = (s_{i\ominus 1}, -m) \in E_{i\ominus 1}^-$. From TV, we can compute two sets V_c and V_s whose elements are in the form of $\vec{\mathbf{v}} = (v_1, v_2, \ldots, v_n)$ and $\vec{\mathbf{v}} \in TV$. For each $\vec{\mathbf{v}} \in V_c$, either $\forall\, i \in [1..n] : v_i \in E_i^-$ or $\forall\, i \in [1..n] : v_i \in E_i^+$. In this case, $\vec{\mathbf{v}}$ is called a *concurrency* vector in S. For each $\vec{\mathbf{v}} \in V_s$, there is at least one *send-receive pair* $(v_i, v_{i\oplus 1})$ in which $v_i \in E_i^-$ and $v_{i\oplus 1} \in E_{i\oplus 1}^+ \cup E_{i\oplus 1}^{++}$. On the other hand, if v_i is not in a send-receive pair, then $v_i = \lambda$. In this case, $\vec{\mathbf{v}}$ is called a *synchronization vector* in S. Loosely speaking, in a concurrency vector, either all processes are sending or all processes are receiving; in a synchronization vector, some processes are grouped into send-receive pairs while the rest do not progress at all. For details, please refer to [7, 8].

To cope with internal transitions, we let $E_i^\mu = \{\mu(s_i)\}$ if μ is defined at s_i; $E_i^\mu = \emptyset$ otherwise. For each $E_i^\mu \neq \emptyset$, we construct an *internal* vector $\vec{\mathbf{v}}$ in S as follows: set $v_i = \mu(s_i)$ and set $v_j = \lambda$ for each $j \neq i$. Denote V_μ as the set of internal vectors in S. Let $V = V_c \cup V_s \cup V_\mu$. Each $\vec{\mathbf{v}} \in V$ is called the *fair progress vector* in S.

The *fair reachability* relation is defined as follows. Given two states $S = (s_1, s_2, \ldots, s_n, c_{n1}, c_{12}, \ldots, c_{n-1n})$ and $S' = (s'_1, s'_2, \ldots, s'_n, c'_{n1}, c'_{12}, \ldots, c'_{n-1n})$, $S \mapsto_f S'$ iff $\exists \vec{\mathbf{v}} \in V(S)$ that leads the system from S to S'. There are four cases to consider:

(1) $\vec{\mathbf{v}} \in V_s(S)$. For each send-receive pair $(v_i, v_{i\oplus 1}), i \in [1..n]$, there are two subcases to consider: (a) $c_{ii\oplus 1} = \epsilon$. Let $v_i = \tau(s_i, -m)$ and $v_{i\oplus 1} = \tau(s_{i\oplus 1}, +m)$. Execution of $(v_i, v_{i\oplus 1})$ will cause transition $\tau(s_i, -m)$ to be taken, followed by transition $\tau(s_{i\oplus 1}, +m)$, where $s'_i \in \tau(s_i, -m)$ and $s'_{i\oplus 1} \in \tau(s_{i\oplus 1}, +m)$. (b) $c_{ii\oplus 1} \neq \epsilon$. Let $v_i = \tau(s_i, -m)$, $v_{i\oplus 1} = \tau(s_{i\oplus 1}, +m')$, and $c_{ii\oplus 1} = m' \cdot c''_{ii\oplus 1}$. Execution of $(v_i, v_{i\oplus 1})$ will cause transitions $\tau(s_i, -m)$ and $\tau(s_{i\oplus 1}, +m')$ to be taken in arbitrary order, where $s'_i \in \tau(s_i, -m)$, $s'_{i\oplus 1} \in \tau(s_{i\oplus 1}, +m')$, and $c'_{ii\oplus 1} = c''_{ii\oplus 1} \cdot m$. Except for the elements affected by the transitions applied in each of the send-receive pairs, all other elements of S' remain the same as those in S.

(2) $\vec{\mathbf{v}} \in V_c(S) \wedge (\forall\, i \in [1..n] : v_i = \tau(s_i, -m_i) \in E_i^-)$. The result of applying $\vec{\mathbf{v}}$ on S is such that $\forall\, i \in [1..n] : s'_i \in \tau(s_i, -m_i)$ and $c'_{ii\oplus 1} = c_{ii\oplus 1} \cdot m_i$.

(3) $\vec{\mathbf{v}} \in V_c(S) \wedge (\forall\, i \in [1..n] : v_i = \tau(s_i, +m_i) \in E_i^+)$. Assume that before applying $\vec{\mathbf{v}}$, $\forall\, i \in [1..n] : c_{i\ominus 1i} = m_i \cdot c''_{i\ominus 1i}$. The result of applying $\vec{\mathbf{v}}$ on S is such that $\forall\, i \in [1..n] : s'_i \in \tau(s_i, +m_i)$ and $c'_{i\ominus 1i} = c''_{i\ominus 1i}$.

(4) $\vec{\mathbf{v}} \in V_\mu(S)$. Suppose $v_i = \mu(s_i)$. The result of applying $\vec{\mathbf{v}}$ on S is such that $\forall\, j \in [1..n] : s'_j \in \mu(s_j)$ if $i = j$; $s'_j = s_j$ otherwise.

Denote $\mapsto_f^*$ as 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 (from S) for s'_i, (m, s'_i), and cycle $\mathcal{C}_i$, respectively. The set of fair reachable states, denoted as $\mathbf{F}$, is called the *fair reachable state space* of P.

Note that $\mapsto_f$ is defined in the same way as that for D-cyclic protocols [7, 8] except for two modifications: First, internal vectors are added into V during fair progress state exploration. Second, due to nondeterminism, the resulting local state s'_i of a transition $\tau(s_i, \sigma)$ ($(\mu(s_i)$) is written as $s'_i \in \tau(s_i, \sigma)$ ($s'_i \in \mu(s_i)$) instead of $s_i = \tau(s_i, \sigma)$ ($s'_i = \mu(s_i)$).

Since $S^0 \in \mathbf{F}$ is a state with equal channel length of zero and any fair progress vector in S^0 maintains the equal channel length property in the resulting state, it is not difficult to show by induction that each fair reachable state is a reachable state with equal channel length. Conversely, suppose S is a reachable state with equal channel length. Let $\{e_1, e_2, \ldots, e_n\}$ be a local execution sequence set from S^0 to S. We construct a *partial fair execution sequence* for S with respect to (w.r.t for short) $\{e_1, e_2, \ldots, e_n\}$, 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 fp is *unique* w.r.t $\{e_1, e_2, \ldots, e_n\}$, although pfs is not always unique. Note that $fp \in \mathbf{F}$. If $fp = S$, then we are done. Suppose $fp(S) \neq S$, then the following lemma holds for fp.

Lemma 3.1 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 and $\tau_k^p \neq \mu(s_k^p)$. (4) $fp \mapsto^* S$ via the remaining transitions from $\{e_1, e_2, \ldots, e_n\}$ in fp.

Based on this lemma, we can show that each reachable state with equal channel length is fair reachable.

Theorem 3.1 $\mathbf{F}$ is exactly the set of reachable states with equal channel length.

We are primarily interested in cyclic protocols whose fair reachable state spaces are finite. In [7], we show that for a D-cyclic protocol, $\mathbf{F}$ 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 3.2 Given a cyclic protocol P without reachable sending cycles. If P is unbounded, then P is simultaneously unbounded.

Lemma 3.3 If a cyclic protocol P is simultaneously unbounded, then its $\mathbf{F}$ is infinite.

Theorem 3.2 Given a cyclic protocol P with a finite $\mathbf{F}$. P is unbounded iff it has a reachable sending cycle.

Theorem 3.3 Given a cyclic protocol P. $\mathbf{F}$ is finite iff P is not simultaneously unbounded.

Theorem 3.4 It is undecidable whether a cyclic protocol P has a finite $\mathbf{F}$.

In [8], we use $\mathcal{P}$ to denote the class of D-cyclic protocols whose $\mathbf{F}$'s are finite. In this paper, we use $\mathcal{Q}$ to denote the class of cyclic protocols whose $\mathbf{F}$'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 $\mathbf{F}$, we mean that it is finite.

4 Fault Coverage of F

As with states in **R**, we define logical errors for states in **F** in a similar way. Give a state $S \in \mathbf{F}$. S is a *fair* deadlock state if it is a deadlock state. S is a *fair* unspecified reception state iff there is a receiving local state $s_i \in S$ such that (i) $c_{i\ominus 1i} = m \cdot c'_{i\ominus 1i}$ and $\tau(s_i, +m)$ is not defined, or (ii) $c_{i\ominus 1i} = \epsilon$, $\tau(s_{i\ominus 1}, -m)$ is defined, and $\tau(s_i, +m)$ is not defined. S is a *fair* unbounded state iff it is in a sending cycle. S is a *fair* indefinite state iff it is in an internal cycle. The set of fair deadlock (unspecified reception, unbounded, indefinite) states is denoted as $\mathbf{F}_{dl}$ ($\mathbf{F}_{ur}, \mathbf{F}_{ub}, \mathbf{F}_{id}$). By Theorem 3.2, a fair unbounded state is well-defined (recall that we assume **F** is finite). Note that a fair unspecified reception state is not an unspecified reception state when $c_{i\ominus 1i} = \epsilon$. Moreover, there might be "dead end" states in **F** whose $V = \emptyset$. However, as for D-cyclic protocols, it can be shown that the notion of fair unspecified reception is sufficient for detecting unspecified receptions in **F** and the occurrence of dead end states does not introduce new types of logical errors in **F** [8].

Let's study the logical error detection capability of **F**. First, notice that all deadlock states are of equal channel length zero. By Theorem 3.1, we have the following result on deadlock detection:

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

However, as for D-cyclic protocols, it is not difficult to see that for detection of logical errors other than deadlock, **F** is not sufficient, and thus finite extension of **F** is needed. Following the same formulation as [8], we reduce the detection of logical errors other than deadlock in $\mathcal{Q}$ to two *local state reachability* problems as follows:

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

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

It should be clear that for $\mathcal{Q}$, if we can solve **P-I**, then we can solve unboundedness and indefiniteness detection; if we can solve **P-II**, then we can solve unspecified reception detection; if we can solve both **P-I** and **P-II**, then we can solve detection of nonexecutable transitions. Although neither **P-I** nor **P-II** is decidable in general (using the results established in [1]), we will show that both of them are decidable for $\mathcal{Q}$ via finite extension of **F**. The line of reasoning is almost identical to that for showing them decidable for $\mathcal{P}$ in [8]. As a result, we will be quite informal in the arguments we make and only highlight the differences along the way. Interested readers should consult [8] for details.

As we have already seen, the need for finite extension in **F** results from the fact that some of the reachable local states are not fair reachable. Therefore, the purpose of finite extension is to uncover those local states. Suppose s_k is reachable but not fair reachable. Then none of the reachable states containing s_k is in **F**. Let S be any reachable state with $s_k \in S$, and $\{e_1, e_2, \ldots, e_n\}$ be a local execution sequence set for S. Let pfs and fp be a partial fair execution sequence and the fair precursor for S w.r.t $\{e_1, e_2, \ldots, e_n\}$, respectively.

By Lemma 3.1, we can find a maximal interval $[i..k]$ in fp such that $\forall j \in [i..k] : |e_j| \neq 0$. Moreover, let τ^p_j be the transition from e_j at s^p_j, then $\tau^p_j \neq \mu(s^p_j)$ and 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_j must take on the transition from e_j if $(j \in [i..k]) \wedge (|e_j| \neq 0)$. Without loss of generality, let's assume that none of the e_j's becomes empty during the construction. Let $\mathbf{F}^{min}_{[i..k]}$ be the set of states from the construction whose sum of the remaining transitions in $\{e_i, e_{i\oplus 1}, \ldots, e_k\}$ is minimum. Note that if $S' \in \mathbf{F}^{min}_{[i..k]}$ and S'' is fair reachable from S' by the construction, then $S'' \in \mathbf{F}^{min}_{[i..k]}$. More importantly, S'' is fair reachable from S' *without progress* in $[i..k]$. Let $\vec{\mathbf{u}}_{[i..k]} = (u_i, u_{i\oplus 1}, \ldots, u_k)$ be the transition vector associated with a state $S' \in \mathbf{F}^{min}_{[i..k]}$, then $\vec{\mathbf{u}}_{[i..k]}$ is a *proper incompatible* transition vector (*pitv*) in S. ($\vec{\mathbf{u}}_{[i..k]}$ is a *pitv* in S iff it satisfies the following four conditions: (1) Each transition in the vector is executable in S and is not an internal transition. (2) S does not have a concurrency vector. (3) There is no send-receive pair in $\vec{\mathbf{u}}_{[i..k]}$. (4) neither u_i nor u_k appears in a send-receive pair in any synchronization vector in S.) In fact, $\vec{\mathbf{u}}_{[i..k]}$ is the same for any S' in $\mathbf{F}^{min}_{[i..k]}$, and thus is a *persistent* proper incompatible transition vector (*ppitv*) in S. (A *pitv* $\vec{\mathbf{u}}_{[i..k]}$ is *persistent* in S iff it is also a *pitv* in any state fair reachable from S without progress in $[i..k]$.) Denote $U_{[i..k]}$ ($W_{[i..k]}$) as the set of *pitv*'s (*ppitv*'s) in S. Notice that although the preceding discussion is based on reachability of s_k, it also applies to the reachability of (m, s_k). To sum up, we have the following lemma:

Lemma 4.1 A local state s_k ((m, s_k)) is reachable but not fair reachable only if there is a state $S' \in \mathbf{F}$ such that $W_{[i..j]} \neq \emptyset$ in S', $k \in [i..j]$, and s_k ((m, s_k)) is reachable from S'.

Therefore, the extension of $\mathbf{F}$ should be based on the set of states in $\mathbf{F}$ whose $W_{[i..j]} \neq \emptyset$ for some interval $[i..j]$. To reduce the cost of extension, we want to compute a *extension set* $\mathbf{F}_T \subseteq \mathbf{F}$ such that its membership can be easily decided and there is a state $S \in \mathbf{F}$ whose $W_{[i..j]} \neq \emptyset$ only if $\mathbf{F}_T \neq \emptyset$.

Let's see how $\mathbf{F}_T$ can be computed. Given a state $S \in \mathbf{F}$ whose $W_{[i..j]} \neq \emptyset$. We construct a graph $\mathbf{FRG}_{[i..j]}$ where S is the initial node and each node in the graph stands for a state fair reachable from S without progress in $[i..j]$. Then we construct the *quotient* graph $\mathbf{QFRG}_{[i..j]}$ such that each node is a strongly connected component (SCC) in $\mathbf{FRG}_{[i..j]}$, denoted as $[S']$, where S' is a state in that SCC. Then $\mathbf{QFRG}_{[i..j]}$ is a directed acyclic graph (DAG). The initial node is denoted as $[S]$. Let TN be the set of terminal nodes in $\mathbf{QFRG}_{[i..j]}$. By definition of *ppitv*, it is clear that $W_{[i..j]}(S') = W_{[i..j]}(S'')$ if $[S'] = [S'']$. Denote $W_{[i..j]}([S'])$ as the set of *ppitv*'s in any state in $[S']$. Then it is also obvious that $W_{[i..j]}(S) = \bigcap_{[S'] \in TN} W_{[i..j]}([S'])$. Since we assume $W_{[i..j]}(S) \neq \emptyset$, it follows that $\forall [S'] \in TN : W_{[i..j]}([S']) \neq \emptyset$. As a result, we only need to focus on those nodes in TN. Given a node $[S'] \in TN$, there are two cases to consider: (1) $[S']$ contains only one state S' but no outgoing edges. (2) There is a fair execution cycle (i.e., a cycle in the corresponding

SCC in $\mathbf{FRG}_{[i..j]}$) among states in $[S']$. Unlike D-cyclic protocols, a fair execution cycle might consist of internal transitions only. In any case, the following lemma shows that $[S']$ contains some error state. Note that $W_{[i..j]} \subseteq U_{[i..j]}$ for any S and $[i..j]$.

Lemma 4.2 Given $S \in \mathbf{F}$ and an interval $[i..j]$. If $U_{[i..j]} \neq \emptyset$ in S and S does not have any fair progress vector without progress in $[i..j]$, then S is a fair unspecified reception state. If S is in a fair execution cycle without progress in $[i..j]$, then S is either a fair unbounded state or a fair indefinite state.

Let $\mathbf{F}_T = \mathbf{F}_{ur} \cup \mathbf{F}_{ub} \cup \mathbf{F}_{id}$. 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 $\mathbf{F}_T$.

Given a state $S \in \mathbf{F}_T$. An interval $[i..j]$ is an *incompatible* interval in S if $U_{[i..j]}(S) \neq \emptyset$. Let IJ be the set of incompatible intervals in S, then $(IJ, \subseteq)$ is a partially odered set. Denote ImJ as the set of maximal elements in $(IJ, \subseteq)$. Our finite extension procedure is based on the finite extension of part of a state S indexed by each $[i..j] \in ImJ$ of S for each $S \in \mathbf{F}_T$. Similar to the approach used for D-cyclic protocols, we can show that such extension can be done in a finite way so that both **P-I** and **P-II** is solvable for S. The formulation of the reachability relation among partial states is the same as that in [8] except for two modifications, as were pointed out in the formulation of fair reachability relation in Section 3. For details, please refer to [8].

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 fair indefinite states into account.

Theorem 4.3 Given a cyclic protocol $P \in \mathcal{Q}$. P has a deadlock iff $F_{dl} \neq \emptyset$. 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 $F_{id} = \emptyset$ 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 offers substantial state reduction over $\mathbf{R}$ but also is very competitive in fault coverage. Furthermore, the decision procedures can be optimized for efficiency in a similar way as for $\mathbf{P}$. We refer the interested readers to [8] for details.

A final remark on indefiniteness is in order here. Given a reachable internal execution cycle $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$, there is a nonempty $\mathcal{C}_i$ that is an internal cycle of P_i. A special case is when each $\mathcal{C}_i$ is an internal cycle of P_i. Suppose $\mathcal{C}$ is such an execution cycle of S, then once P enters state S, each process could execute internal transitions indefinitely without any communications occur in P. In this case, we say P is *globally indefinite*. If we define a function ϕ that marks each normal transition in P_i as "progress" and marks each internal transition in P_i as "nonprogress", then global indefiniteness becomes a special case of *livelock* defined in [4, 5, 7]. In [7], we generalized the results of [4, 5] from $n = 2$ to $n \geq 2$ and showed that livelock is decidable within $\mathbf{F}$ for class $\mathcal{P}$. With the fair reachability notion formulated in Section 3 of this paper, it is not difficult to show that livelock is also decidable for $\mathcal{Q}$ within $\mathbf{F}$. As a result, global indefiniteness is also decidable for $\mathcal{Q}$ without finite extension on $\mathbf{F}$.

5 Discussion

Through this study, it is clear that the adaptation to nondeterminism in the model can be done without much modification in the original formulation. The inclusion of internal transitions, on the other hand, has several implications: First, internal transitions must be incorporated into the formulation of fair progress vectors. In order to ensure that each reachable state with equal channel length is also fair reachable, each internal transition in a state must be executed individually, which leads us to treat normal transitions and internal transitions in a state separately when computing the set of fair progress vectors in that state. Second, the clean separation between normal and internal transitions allows us to adapt our approach in the augmented model with little effort. However, allowing each process to execute only one (internal) transition at a time is inconsistent with the general philosophy of fair progress state exploration. As a result, more redundancy is introduced in the state exploration process due to the interleaving of equivalent execution sequences. Third, the existence of a reachable internal cycle in a process results in a new type of logical error called indefiniteness. Since the set of fair indefinite states is a subset of the extension set, finite extension is in general more costly than the one in [8] for D-cyclic protocols.

A closer look at indefiniteness leads to the following generalization: A cyclic protocol P is *k-indefinite* iff P has a reachable internal execution cycle $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ in which there are at least $k \in [1..n]$ nonempty $\mathcal{C}_i$'s in $\mathcal{C}$. Clearly, indefiniteness and global indefiniteness represent the two ends of the k-indefiniteness spectrum and we know that both of them are decidable for $\mathcal{Q}$. We can also generalize the livelock notion in [4, 5, 7] to *k-livelock* for cyclic protocols in a similar way. We already know that n-livelock is decidable for $\mathcal{Q}$. Moreover, **P-I** being decidable for $\mathcal{Q}$ implies that 1-livelock is also decidable for $\mathcal{Q}$. Now the question is whether k-indefiniteness, or more generally k-livelock, can be decided effectively and efficiently based on $\mathbf{F}$ for cyclic protocols in $\mathcal{Q}$ when $n > 2$ and $1 < k < n$. We are currently working on this issue.

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, 3] 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 their 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 handle internal cycles, it is not clear why such restriction was made in their formulation. Second, the "parallelwise" condition imposed on reduced transitions 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 much 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.

6 Conclusion

In this paper, we extended the generalized fair reachability analysis technique to cyclic protocols with nondeterminism and internal transitions. We showed that most of the results established for cyclic protocols without nondeterminism and internal transitions in [7, 8] can be carried over to cyclic protocols with nondeterminism 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 nondeterminism and internal transitions are allowed.

As for future work, we are going to address the following issues: (1) k-indefiniteness and

k-livelock; (2) More general and yet regular protocol communications topologies; and (3) Other formal models such as the extended finite state machines model.

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] L. Cacciari and O. Rafiq, "Decidability Issues in Reduced Reachability Analysis," ICNP'93, San Francisco, CA, October 19–22, 1993, pp. 158–165.

[4] 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.

[5] M.G. Gouda, C.H. Chow, and S.S. Lam, "On the Decidability of Livelock Detection in Networks of Communicating Finite State Machines," PSTV'85, Y. Yemini, R. Strom, and S. Yemini (Ed.), 1985, pp. 47–56.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

Appendix: Proofs of Lemmas and Theorems

Lemma 3.1 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 and $\tau_k^p \neq \mu(s_k^p)$. (4) $fp \mapsto^* S$ via the remaining transitions from $\{e_1, e_2, \ldots, e_n\}$ in fp.

Proof: $\{e_1, e_2, \ldots, e_n\}$ being a local execution sequence set for S implies that there is no deadlock or unspecified reception during the execution of transitions in $\{e_1, e_2, \ldots, e_n\}$ from S^0 to S. Since $fp \neq S$, there must be some transitions in $\{e_1, e_2, \ldots, e_n\}$ remained to be executed in fp, i.e., $\exists k \in [1..n] : |e_k| \neq 0$. Thus, (1) holds. Suppose $|e_k| \neq 0$, then τ_i^p cannot be an internal transition, otherwise an internal vector can be found in fp. If τ_k^p is a sending transition, then it is executable. Hence, τ_k^p is not executable only if it is a receiving transition and $c_{k\ominus 1k}^p = \epsilon$, otherwise there will be an unspecified reception along $\{e_1, e_2, \ldots, e_n\}$. In this case, there must be at least one such $\tau_i^p, |e_i| \neq 0$, that is a sending transition; otherwise the protocol cannot precede beyond fp to reach S. As a result, a send-receive pair can be derived from the transitions in fp, which contradicts the assumption that no fair progress vector can be derived from fp based on the remaining transitions in $\{e_1, e_2, \ldots, e_n\}$. Therefore, if $|e_k| \neq 0$, then τ_k^p must be executable, i.e., (3) holds. Suppose now $\forall\, i \in [1..n] : |e_i| \neq 0$, then each τ_i^p is executable. As a result, either a concurrency vector or a synchronization vector can be derived from $\vec{\mathbf{t}} = (\tau_1^p, \tau_2^p, \ldots, \tau_n^p)$, which also contradicts the assumption that no fair progress vector can be derived from fp based on the remaining transitions in $\{e_1, e_2, \ldots, e_n\}$. Thus, $\exists\, k \in [1..n] : |e_k| = 0$, i.e., (2) holds. Finally, by induction on the number of remaining transitions in $\{e_1, e_2, \ldots, e_n\}$ from fp to S, it is obvious that S is reachable from fp via those remaining transitions, i.e., (5) holds. ∎

Theorem 3.1 **F** is exactly the set of reachable state with equal channel length.

Proof: We need to show that S is fair reachable iff it is a reachable state with equal channel length.

(*Only If:*) Suppose S is fair reachable. Then S is reachable. Let fs be a fair execution sequence for S. Denote $fs = X^0 \xrightarrow{\vec{\mathbf{v}}_1} X^1 \xrightarrow{\vec{\mathbf{v}}_2} \cdots \xrightarrow{\vec{\mathbf{v}}_k} X^k, k \geq 0$, where $X^0 = S^0$, $\forall\, j \in [1..k]$: $X^{j-1} \mapsto_f X^j$ via fair progress vector $\vec{\mathbf{v}}_j$, and $X^k = S$. We claim that S is of equal channel length by induction on k.

Basis: $k = 0$. In this case, $S = S^0$. The claim holds trivially.

Induction: Suppose S is of equal channel length for $k = k' \geq 0$. We want to show for $k = k' + 1$. Note that X^{k-1} is fair reachable via a fair execution sequence of length k'. By induction hypothesis, X^{k-1} is of equal channel length. Now, $X^{k-1} \mapsto_f S$ via fair progress vector $\vec{\mathbf{v}}_k$. If $\vec{\mathbf{v}}_k$ is a concurrency vector, then it will either increase each channel length by one or decrease each channel length by one when applied to X^{k-1}. If $\vec{\mathbf{v}}_k$ is a synchronization vector or an internal vector, then it will not change the length of any channel when applied to X^{j-1}. Hence, S is also of equal channel length. The claim holds for $k = k' + 1$.

Therefore, S is a reachable state with equal channel length.

(*If:*) Suppose S is a reachable state with equal channel length $K \geq 0$. We want to show that S is fair reachable. Let $\{e_1, e_2, \ldots, e_n\}$ be a local execution sequence set for S and fp be the fair precursor of S w.r.t $\{e_1, e_2, \ldots, e_n\}$. Then fp is fair reachable. From the preceding argument, fp is of equal channel length. Let K' be the channel length in fp. Let $[i..j]$ be an interval in fp such that $\forall\, k \in [i..j] : |e_k| \neq 0$ and $|e_{i\ominus 1}| = |e_{j\oplus 1}| = 0$. By Lemma 3.1, such an interval exists. Moreover, $\forall\, k \in [i..j] : \tau_k^p \neq \mu(s_k^p)$ and is executable in fp. Note that in this case, either τ_i^p is a receiving transition or τ_j^p is a sending transition. Otherwise, a send-receive pair can be derived from $(\tau_i^p, \tau_{i\oplus 1}^p, \ldots, \tau_j^p)$, which contradicts the assumption that no fair progress vector can be derived from fp. There are three cases to consider:

(1) $K' < K$. Note that the length of channel $C_{i\ominus 1 i}$ cannot be increased. By the time the protocol gets to S, the length of channel $C_{i\ominus 1 i}$ will be less than K.

(2) $K' > K$. Note that the length of channel $C_{jj\oplus 1}$ cannot be decreased. By the time the protocol gets to S, the length of channel $C_{jj\oplus 1}$ will be greater than K.

(3) $K' = K$. There are two subcases to consider:

(a) τ_i^p is a receiving transition. Then after the execution of τ_i^p, the length of channel $C_{i\ominus 1 i}$ will be $K - 1$. Note that the length of channel $C_{i\ominus 1 i}$ cannot be increased. By the time the protocol gets to S, the length of channel $C_{i\ominus 1 i}$ will be no greater than $K - 1$.

(b) τ_j^p is a sending transition. Then after the execution of τ_j^p, the length of channel $C_{jj\oplus 1}$ will be $K + 1$. Note that the length of channel cannot be decreased. By the time the protocol gets to S, the length of channel will be no less than $K + 1$.

In all cases, there will be a channel whose length is not K when the protocol gets to S, which contradicts the assumption that S is of equal channel length K. Hence, S is fair reachable. ∎

Lemma 3.2 Given a cyclic protocol P without reachable sending cycles. If P is unbounded, then P is simultaneously unbounded.

Proof: Since P is unbounded, P has at least one unbounded channel. Without loss of generality, suppose channel C_{12} is unbounded.

Since C_{12} is unbounded, there must exist an infinite execution sequence $e = \{e_1, e_2, \ldots, e_n\}$ such that for any $k \geq 0$, there is a state reachable via a prefix of e such that $|c_{12}| > K$. Moreover, since each process P_i has no reachable sending cycles, each e_i is composed of infinitely many sends and receives, and there can only be at most $|\mathbf{S}_i| - 1$ consecutive receives before a send in e_i, where $|\mathbf{S}_i|$ is the number of states in P_i. As a result, there must be at least one such execution sequence along which P can proceed indefinitely, i.e., no unspecified reception can occur along this sequence, otherwise C_{12} will be bounded. Fix $e = \{e_1, e_2, \ldots, e_n\}$ as such an execution sequence.

Define a function $f : [0..n-1] \rightarrow \mathcal{N}$, $\mathcal{N}$ being the set of natural numbers, as follows:

$$f(i) = \begin{cases} 1 & \text{if } i = 0 \\ 1 + |S_{n \ominus (i \ominus 1)}| \times f(i \ominus 1) & \text{if } 0 < i < n \end{cases}$$

Based on the preceding argument, for any $K \geq 0$, there is a state $S = (s_1, s_2, \ldots, s_n, c_{n1}, c_{12}, \ldots, c_{n-1n})$ reachable via a prefix of e such that $|c_{12}| = f(n \ominus 1) \times K'$, where $K' > K$. If all other channels have more than K messages, we are done. Suppose not, starting from S, in the order from P_2 to P_n, each process $P_i, i \in [2..n]$, can receive $|\mathbf{S}_i| \times f(n \ominus i) \times K'$ messages from channel $C_{i \ominus 1 i}$, and as a result, send at least $f(n \ominus j)$ messages to channel $C_{ii \oplus 1}$. In the end, the protocol must arrive at a reachable state such that each channel should have at least K' messages. Therefore, there is a reachable global state in which each channel length is greater than K, i.e., P is simultaneously unbounded. ∎

Lemma 3.3 If a cyclic protocol P is simultaneously unbounded, then its $\mathbf{F}$ is infinite.

Proof: We first show the following: if there is a reachable state $S = (s_1, s_2, \ldots, s_n, c_{n1}, c_{12}, \ldots, c_{n-1n})$ such that $\forall\, i \in [1..n] : |c_{ii\oplus 1}| \geq K$ for some constant $K \geq 0$, then there exists a fair reachable state $S' = (s'_1, s'_2, \ldots, s'_n, c'_{n1}, c'_{12}, \ldots, c'_{n-1n})$ such that $\forall\, i \in [1..n] : |c'_{ii\oplus 1}| \geq K$.

> First, if $S \in \mathbf{F}$, then let $S' = S$, we are done. Second, if $K = 0$, then let $S' = S^0$, and we are done. Now suppose $S \notin \mathbf{F}$ and $K > 0$. Let $e = \{e_1, e_2, \ldots, e_n\}$ be an execution sequence for S. Based on $\{e_1, e_2, \ldots, e_n\}$, we construct the partial fair execution sequence for S to get to fp, the fair precursor of S. Clearly, $fp \in \mathbf{F}$ and is of equal channel length by Theorem 3.1. Suppose fp is of channel length K'. If $K' \geq K$, then let $S' = fp$, and we are done. Suppose not, by Lemma 3.1, $\exists k \in [1..n] : |e_k| = 0$. Note that from state fp and on, the length of channel $C_{kk\oplus 1}$ cannot be increased with the execution of remaining transitions in e by other processes. Therefore, at the end of the execution of e, i.e., in state S, the length of channel $C_{kk\oplus 1}$ will be less than K, which contradicts the fact that every channel length in S is no less than K. Hence, fp must have channel length no less than K. In summary, we can find a fair reachable state whose channel length is no less than K.

Now since P is simultaneously unbounded, $\forall\, K \geq 0$, there exists a $K' > K$ such that there is a reachable state S in which each channel length is no less than K'. As a result, there is a fair reachable state S' whose channel length is no less than K'. Therefore, $\mathbf{F}$ is infinite. ∎

Theorem 3.2 Given a cyclic protocol P with a finite $\mathbf{F}$. P is unbounded iff it has a reachable sending cycle.

Proof: Obviously, if P has a reachable sending cycle, then P is unbounded. Suppose P is unbounded but does not have a reachable sending cycle. Then By Lemma 3.2, P is simultaneously unbounded. By Lemma 3.3, $\mathbf{F}$ is infinite. A contradiction. ∎

Theorem 3.3 Given a cyclic protocol P. $\mathbf{F}$ is finite iff P is not simultaneously unbounded.

Proof: Suppose $\mathbf{F}$ is infinite, then $\mathbf{F} = \bigcup_{k=0}^{\infty} \mathbf{F}_k$ is infinite. Thus, $\forall K \geq 0\ \exists K' > K$: $\mathbf{F}_{K'} \neq \emptyset$. Since any state in $\mathbf{F}$ is of equal channel length, P is simultaneously unbounded. On the other hand, by Lemma 3.3, if P is simultaneously unbounded, then $\mathbf{F}$ is infinite. ∎

Theorem 3.4 It is undecidable whether a cyclic protocol P has a finite $\mathbf{F}$.

Proof: Since it is undecidable whether a D-cyclic protocol has a finite fair reachable state space [7], it follows that it is also undecidable whether a (general) cyclic protocol P has a finite $\mathbf{F}$. ∎

Lemma 4.2 Given $S \in \mathbf{F}$ and an interval $[i..j]$. If $U_{[i..j]} \neq \emptyset$ in S and S does not have any fair progress vector without progress in $[i..j]$, then S is a fair unspecified reception state. If S is in a fair execution cycle without progress in $[i..j]$, then S is either a fair unbounded state or a fair indeterminate state.

Proof: By definition, $U_{[i..j]} \neq \emptyset$ in S implies that $\forall\, k \in [i..j] : E_k \neq \emptyset$ in S. Thus S is not a deadlock state. Denote $\overline{[i..j]}$ as the complement interval of $[i..j]$ w.r.t $[1..n]$. Suppose S is not a fair unspecified reception state. Then $\forall\, k \in \overline{[i..j]} : E_k \cup E_k^{++} \cup E_k^{\mu} \neq \emptyset$. As a result, a fair progress vector can be derived from each $\vec{\mathbf{t}} \in TV$. Let $\vec{\mathbf{u}}_{[i..j]}$ be a *pitv* in S. Let $\vec{\mathbf{t}}$ be a pseudo transition vector in TV such that $\forall\, k \in [i..j] : u_k = t_k$. Then a fair progress vector $\vec{\mathbf{v}}$ can be derived from $\vec{\mathbf{t}}$ and $\forall\, k \in [i..j] : v_k = \lambda$. Hence $\vec{\mathbf{v}}$ is a fair progress vector in S without progress in $[i..j]$. A contradiction. Therefore, S is a fair unspecified reception state.

Now suppose S is in a fair execution cycle fc without progress in $[i..j]$. Let $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ be the corresponding local execution cycle set of S. Then $\forall\, k \in [i..j] : \mathcal{C}_k$ is empty. Since fc is not empty, there must be a nonempty interval $[h..l]$ in S such that $\{i..j\} \cap \{h..l\} = \emptyset$, $\forall\, k \in [h..l] : \mathcal{C}_k$ is nonempty, and $\mathcal{C}_{h\ominus 1}$ and $\mathcal{C}_{l\oplus 1}$ are empty. If there is a $\mathcal{C}_k, k \in [h..l]$, that is an internal cycle of P_k, then S is an indeterminate state. Otherwise, we claim that $\mathcal{C}_h$ is a sending cycle in P_h. Suppose not. Then there is at least one receiving transition in $\mathcal{C}_h$. Assume S is of channel length K. Then going through fc once will decrease the length of channel $C_{h\ominus 1h}$ by one. On the other hand, $P_{h\ominus 1}$ is idle during the execution of fc. As a result, executing fc once will not lead the system back to S, contradicting the assumption that fc is a fair execution cycle. Therefore, $\mathcal{C}_h$ must be a sending cycle in P_h, i.e., S must be a fair unbounded state. ∎

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

Proof: The proof requires the formulation of partial state reachability and the finite extension construction of partial states, both of which are omitted in this paper due to space limitations. Please refer to [8] for details. ∎

Theorem 4.3 Given a cyclic protocol $P \in \mathcal{Q}$. P has a deadlock iff $F_{dl} \neq \emptyset$. 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 indeterminate but $F_{id} = \emptyset$ 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.

Proof: The deadlock case is obvious from Theorem 3.1. Suppose P has an unspecified reception but $\mathbf{F}_{ur} = \emptyset$. Then there is a reachable state S such that $(m, s_k) \in S$, s_k is local receiving state, and $\tau(s_k, +m)$ is not defined. Since $\mathbf{F}_{ur} = \emptyset$, (m, s_k) is reachable but not fair reachable. By Lemma 4.1 and Lemma 4.2, $\mathbf{F}_T \neq \emptyset$. Since $\mathbf{F}_T = \mathbf{F}_{ur} \cup \mathbf{F}_{ub} \cup \mathbf{F}_{id}$, we must have $\mathbf{F}_{ub} \cup \mathbf{F}_{id} \neq \emptyset$. The proofs for unboundedness, indeterminacy, and nonexecutable transition can be carried out in a similar way.

Now suppose P is logically correct, then there is no reachable error states in $\mathbf{F}$. Conversely, if $\mathbf{F}$ is free of logical errors, then $\mathbf{F}_T = \emptyset$. P cannot have a deadlock since all deadlock states are included in $\mathbf{F}$. Based on the discussion in the preceding paragraph, P cannot have any other logical errors either since otherwise we will have $\mathbf{F}_T \neq \emptyset$. Hence, P is logically correct. ∎