Generalized Fair Reachability Analysis for Cyclic Protocols*

Hong Liu Raymond E. Miller

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

Abstract

In this paper, the notion of fair reachability is generalized to cyclic protocols with $n \geq 2$ machines. Substantial state reduction can be achieved via fair progress state exploration. It is shown that the fair reachable state space is exactly the set of reachable states with equal channel length. As a result, deadlock detection is decidable for $\mathcal{P}$, the class of cyclic protocols whose fair reachable state spaces are finite. The concept of simultaneous unboundedness is defined and the lack of it is shown to be a necessary and sufficient condition for a protocol to be in $\mathcal{P}$. Through finite extension of the fair reachable state space, it is also shown that detection of unspecified receptions, unboundedness, and nonexecutable transitions are all decidable for $\mathcal{P}$. Furthermore, it is shown that any protocol $\mathcal{P}$ is logically correct if and only if there is no logical error in its fair reachable state space. This study shows that for the class $\mathcal{P}$, our generalized fair reachability analysis technique not only achieves substantial state reduction but also maintains very competitive logical error coverage. Therefore, it is a very useful technique to prove logical correctness for a wide variety of cyclic protocols.

1 Introduction

One of the most popular models for protocol specification and validation is the *communicating finite state machine* model. In this model, processes are modeled as finite state machines communicating with each other via *FIFO* channels. *Reachability analysis* can be employed to systematically explore the entire protocol state space to validate the logical correctness of a protocol against some common errors, such as deadlocks, unspecified receptions, unboundedness, and nonexecutable transitions. However, for general protocols, finding out whether a logical error exists is not always decidable [1]. Furthermore, even when decidability is ensured, the explosion of state space during reachability analysis renders its use impractical for most real world protocols. As a result, much of the research has been devoted to identifying the class of protocols with decidable logical errors and devising state reduction techniques to overcome the state explosion problem during state space exploration. For a survey of these methods, please refer to [17].

Fair reachability analysis was proposed as one of the improved reachability analysis techniques for protocols with two machines [16, 5]. In fair reachability analysis, two machines are forced to make progress at the same time whenever possible. State reduction is achieved by only generating those fair progress states. More importantly, if the reduced state space is finite, logical correctness of a protocol can be decided, although in some cases, finite extension

*Research reported in this paper was supported by NASA Center of Excellence in Space Data and Information Sciences Under USRA Subcontract No. 550-66 and by NASA Grant No. NAG 5-2648.

of the reduced state space is necessary [5]. However, the concept of fair reachability and its effectiveness for general protocols with more than two machines have not yet been studied. To fill this gap, we investigate the generalization of this technique to cyclic protocols with $n \geq 2$ machines. Through the study, its effectiveness for cyclic protocol validation is shown.

The rest of the paper is organized as follows. In the following section, we briefly review previous research on fair reachability analysis and cyclic protocols, and highlight the results presented in this paper. Then the communicating finite state machine model is introduced. In Section 4, we generalize the fair reachability notion for cyclic protocols and study the basic properties of fair reachable state space. It is shown that for the class of cyclic protocols with finite fair reachable state spaces, deadlock detection is decidable; however, for detection of other logical errors, fair reachable state space is not sufficient. In Section 5, we show how finite extension can be performed on a finite fair reachable space so that logical errors other than deadlock can be detected effectively and efficiently. We summarize the paper with open problems in Section 6. The proofs of some lemmas and theorems in this paper are given in the appendix.

2 Previous Work

Fair reachability analysis was proposed as a strategy for reducing state explosion during validation of protocols modeled as two communicating finite state machines. Rubin and West first observed the redundancy of state exploration in reachability analysis due to equivalent sequences of interleaving transitions [16]. Based on this observation, they proposed a canonical sequence technique that forces the two machines to progress at the same speed during state exploration. They reported a large percentage reduction in state generation when this technique was incorporated into reachability analysis. In [5], Gouda and Han named this technique *fair reachability analysis*. For protocols whose fair reachable state spaces are finite, detection of deadlock and unspecified reception were shown to be decidable in [16], while detection of boundedness was proved to be decidable in [5]. Gouda et al also showed that if one of the channels is bounded, then the protocol has a finite fair reachable state space [4].

Recently, Cacciari and Rafiq extended the above idea to protocols with "internal" transitions using a technique called *reduced reachability analysis* [2]. In their approach, two machines are allowed to proceed at the same time only if the *parallelwise* condition is satisfied. They showed that detection of deadlock and unspecified reception are decidable for protocols whose reduced reachable state spaces are finite. However, it is not clear under what conditions a protocol can have a finite reduced reachable state space.

One important aspect about fair reachability analysis is that in each fair reachable state, the length of each channel is equal [16, 4]. We call this property the *equal channel length* property of fair reachable state space. A reduced reachable state space generated in [2] does not always have this property. This is, we feel, one of the major reasons that makes it more difficult to find a (sufficient) condition for the class of protocols with finite reduced reachable state spaces.

Fair reachability analysis is of importance not only because it can reduce the number of global states explored, but also because it has the capability to handle some protocols with unbounded channels [5]. Although in [16], the authors claimed to extend this technique to protocols with $n \geq 2$ communicating finite state machines, so far, we have not seen any follow-up reports on this issue.

It should be noted that for bounded protocols, the classic reachability technique can be used for protocols with $n \geq 2$ communicating finite state machines. But research in analysis of protocols with unbounded channels has been mostly limited to only cyclic protocols [11, 12, 14, 15]. Jan Pachl is probably the first person who formalized and investigated the class of cyclic protocols, though many of his important results are contained in his unpublished research report [11]. His method is based on the channel expression concept. In [11], he showed that the detection of deadlock and unspecified reception are decidable for the class of cyclic protocols with one channel whose channel expressions are recognizable. However, he wrote in [11] that the decision procedure is hopelessly inefficient for any practical purpose.

In [14], Peng and Purushothaman showed that for the class of cyclic protocols with exactly one unbounded channel, deadlock detection problem is decidable. Their method relied on the construction of a "stable cover set" and the construction of a finite automaton to recognize the stable cover set. It is not clear, however, whether this procedure can be automated efficiently. In [15], they proposed a data flow approach to analyzing deadlock and unspecified reception for a protocol with $n \geq 2$ machines by computing a superset of the set of reachable states as an approximate solution for a set of data flow equations. While this approach works for general protocols, the results of the analysis are incomplete. It is unknown for what class of protocols the data flow analysis can yield an exact solution. Furthermore, this approach also suffers from state explosion, as stated by the authors in [15].

In summary, for the analysis of cyclic protocols with $n \geq 2$ communicating finite state machines, only the decidability aspect has been studied. The complexity of decision procedures has been largely ignored. For practical analysis, it is highly desirable that the decision procedure be efficient. Moreover, none of these techniques were targeted to the detection of unboundedness and nonexecutable transitions. In addition, all the methods proposed for cyclic protocol validation analyze global states from the channel language perspective [10]. Reachability analysis, which has been a main focus in the analysis of protocols with two machines, has not been fully integrated into any of these approaches. As a matter of fact, it seems that there is a gap between protocols with two machines and protocols with more than two machines. Most of the methods, if not all, that have been proposed for the two machine case have not yet been carried over to the $n \geq 2$ case.

In this paper, we bridge this gap by looking into the possibility of applying the fair reachability technique to the validation for cyclic protocols with $n \geq 2$ communicating finite state machines. This study produces many interesting new results. Our contributions in this paper are summarized as follows: First, we show that the set of fair reachable states is exactly the set of reachable states with equal channel length. As a result, deadlock detection is decidable for the class of cyclic protocols whose fair reachable state spaces are finite. Second, we show that the fair reachable state space of a cyclic protocol is finite if and only if (iff for short) the channels of the protocol are *not simultaneously unbounded*. For the first time, the class of cyclic protocols with finite fair reachability graphs can now be exactly characterized. Even for the $n = 2$ case, this condition is weaker than the one in [5]. For completeness, we also show that this condition is undecidable for cyclic protocols. Third, for logical errors other than deadlock, we show how a finite fair reachable state space can be finitely extended so that these errors can be detected effectively and efficiently. As a result, all logical errors are decidable for the class of cyclic protocols with finite fair reachable state spaces. During the study, we also discover a complete characterization of fair reachable state space in terms of logical error coverage. Fourth, regarding the class of cyclic protocols whose deadlock and unspecified reception detection are decidable, for $n = 2$, our result properly includes the ones studied in [16, 5]; for $n \geq 2$, our result properly contains the one examined in [14] and complements the ones investigated in [15, 11, 12]. More importantly, our decision procedure is much more straightforward and efficient for practical analysis, which was lacking in both [14, 15] and [11, 12]. Furthermore, we also show the decidability of unboundedness and nonexecutable transition detection for the class of cyclic protocols with finite fair reachable state spaces, which are not addressed in any previous approaches except the one in [5] for unboundedness in the $n = 2$ case.

Generalized fair reachability analysis for cyclic protocols was first reported in [7], along with the decidability result of deadlock detection for the class of cyclic protocols with finite fair reachability graphs. Then, the fair reachability notion was revised to achieve further state reduction and allow for easier proofs. The results on basic formulation and deadlock detection were given in PSTV'94 [8], while the results on detection of other logical errors were presented in ICNP'94 [9]. This paper is the combination of results in [8] and [9] with a few modifications.

3 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. 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 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*: $\bigcup_{i=1}^{n}(\mathbf{S}_i \times \tilde{M}_i \rightarrow \mathbf{S}_i)$. 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 $\sigma = -m$ ($\sigma = +m$). A transition cycle $\mathcal{C}_i$ in P_i is a cycle in the transition graph of P_i. It is a *sending* (*receiving*) cycle in P_i iff all the transitions in $\mathcal{C}_i$ are sending (receiving) transitions. s_i is a *sending* (*receiving*) local state iff all transitions defined in s_i are sending (receiving) transitions. We use the notation $\tau' = \tau(s_i, \sigma)$ to give a name τ' for this transition, and use the notation $s_i' = \tau(s_i, \sigma)$ to denote that s_i' is the local state resulting from the execution of the transition. By definition, each P_i is deterministic but partially defined.

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.

Example: A cyclic protocol with four processes is depicted in Figure 1. This protocol will be used as the example throughout this paper.

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

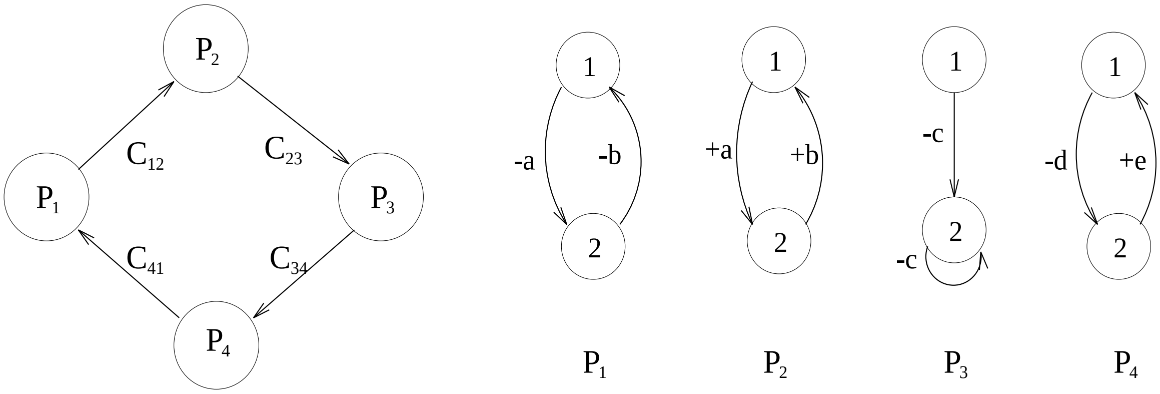

Figure 1: A Cyclic Protocol with 4 Machines

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 $C_{i\ominus 1i}$ 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 letters s_i, x_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) $s'_i = \tau(s_i, -m)$ and $c'_{ii\oplus 1} = c_{ii\oplus 1} \cdot m$, (2) $s'_i = \tau(s_i, +m)$ and $c_{i\ominus 1i} = m \cdot c'_{i\ominus 1i}$. 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) iff there is a state S' such that $S^0 \mapsto^* S'$ ($S \mapsto^* S'$) and $(m, s'_i) \in S'$. A sending 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).

Given a reachable state S. S is a *deadlock* state iff it is a receiving state and each channel is empty. S has an *unspecified reception* iff there is a receiving local state $s_i \in S$ such that $c_{i\ominus 1i} = m \cdot c'_{i\ominus 1i}$ and $\tau(s_i, +m)$ is not defined. A transition $\tau(s_i, \sigma)$ defined at s_i is *executable* if there is a reachable state S such that (i) $\sigma = -m$ and $s_i \in S$, or (ii) $\sigma = +m$ and $(m, s_i) \in S$; otherwise it is *nonexecutable*. P is *unbounded* iff for each $K \geq 0$ there is a reachable state in which there is a channel whose length is greater than K. Deadlock, unspecified reception, nonexecutable transition, and unboundedness are called *logical errors*. P is *logically correct* iff $\mathbf{R}$ is free of logical errors. For protocol validation, we check states in $\mathbf{R}$ against logical errors. This state exploration technique is called *reachability analysis*. However, it was shown in [1] that even for $n = 2$, none of the logical errors is decidable for (cyclic) protocols. As a result, none of the logical errors is decidable for cyclic protocols in general. For completeness, we list this result as a theorem below.

Theorem 3.1 Detection of deadlock, unspecified reception, nonexecutable transition and unboundedness are all undecidable for cyclic protocols.

4 Generalized Fair Reachability Analysis

In this section, we generalize the fair reachability notion to cyclic protocols with $n \geq 2$ machines by incorporating the concepts of synchronization and concurrency into the formulation of fair progress vectors. With that, we show that the set of fair reachable states is exactly the set of reachable states with equal channel length. Then, we establish a necessary and sufficient condition for a cyclic protocol to have a finite fair reachable state space. We also study the logical error detection capability of fair reachable state space. For conciseness, we use "fair reachability" for "generalized fair reachability" from now on.

4.1 Basic Formulation

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. The set of all executable transitions at s_i in S is denoted as $E_i = E_i^- \cup E_i^+$, where E_i^- and E_i^+ stand for the set of executable sending and receiving transitions at s_i in S, respectively. We also denote E_i^{++} as the set of enabled transitions at s_i in S. ($\tau(s_i, \sigma)$ is *enabled* iff $\sigma = +m$, $c_{i\ominus 1 i} = \epsilon$, and $\tau(s_{i\ominus 1}, -m)$ is defined.)

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.

Given a state S and an interval $[i..j]$. A *pseudo transition vector* in S is a tuple $\vec{\mathbf{t}}_{[i..j]} = (t_i, t_{i\oplus 1}, \ldots, t_j)$ such that $\forall\, k \in [i..j] : t_k \in E_k \cup E_k^{++} \cup \{\lambda\}$, where λ stands for a *null* transition in P_k. $\vec{\mathbf{t}}_{[i..j]}$ is a *transition vector* in S iff $\forall\, i \in [1..n] : E_i \neq \emptyset$ and $\forall\, i \in [1..n] : t_i \in E_i$. We drop $[i..j]$ from the notation when $\{i..j\} = \{1..n\}$.

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 $t_{i\ominus 1} = (s_{i\ominus 1}, -m) \in E_{i\ominus 1}^-$. For each pseudo transition vector $\vec{\mathbf{t}} \in TV$, we compute a pseudo transition vector $\vec{\mathbf{v}} = (v_1, v_2, \ldots, v_n)$ from $\vec{\mathbf{t}}$ according to one of the following three cases:

(1) $\vec{\mathbf{t}} \in (\times_{k=1}^n E_k^-) \cup (\times_{k=1}^n E_k^+)$. In this case, set $\vec{\mathbf{v}} = \vec{\mathbf{t}}$. In other words, $\vec{\mathbf{v}}$ has either all processes sending or all processes receiving. $\vec{\mathbf{v}}$ is called a *concurrency vector* in S.

(2) $\exists j : (t_j \in E_j^-) \wedge (t_{j\oplus 1} \in E_{j\oplus 1}^+ \cup E_{j\oplus 1}^{++})$. $(t_{i\ominus 1}, t_i)$ is called a *send-receive pair* in $\vec{\mathbf{t}}$. For each $i \in [1..n]$, if $((t_i \in E_i^-) \wedge (t_{i\oplus 1} \in E_{i\oplus 1}^+ \cup E_{i\oplus 1}^{++})) \vee ((t_{i\ominus 1} \in E_{i\ominus 1}^-) \wedge (t_i \in E_i^+ \cup E_i^{++}))$, then set $v_i = t_i$; else set $v_i = \lambda$. In other words, $\vec{\mathbf{v}}$ retains all the send-receive pairs in $\vec{\mathbf{t}}$. $\vec{\mathbf{v}}$ is called a *synchronization vector* in S.

(3) Neither condition (1) nor condition (2) holds. In this case, set each $v_i = \lambda$. The resulting pseudo transition vector is called the *null vector*, indicating no progress from any process P_i in S.

For each $\vec{\mathbf{v}}$ thus computed, $\vec{\mathbf{v}}$ is a *fair progress vector* in S iff it is either a concurrency vector or a synchronization vector in S. Denote V_c (V_s) as the set of concurrency (synchronization) vectors in S. Let $V = V_c \cup V_s$. V is called the *fair progress vector space* in S. It should be clear that if $\forall\, i \in [1..n] : E_i \cup E_i^{++} \neq \emptyset$, then a fair progress vector can be derived from each $\vec{\mathbf{t}} \in TV$.

The *fair reachability* relation can be 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 three cases to consider:

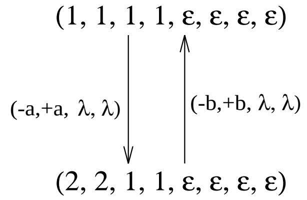

Figure 2: Fair Reachability Graph

(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 $s'_i = \tau(s_i, -m)$ and $s'_{i\oplus 1} = \tau(s_{i\oplus 1}, +m)$.

 (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 $s'_i = \tau(s_i, -m)$, $s'_{i\oplus 1} = \tau(s_{i\oplus 1}, +m')$, 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 $\forall\, i \in [1..n] : s'_i = \tau(s_i, -m_i)$ 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 $\vec{\mathbf{v}}$, $\forall\, i \in [1..n] : c_{i\ominus 1 i} = m_i \cdot c''_{i\ominus 1 i}$. The result of applying $\vec{\mathbf{v}}$ on S is such that $\forall\, i \in [1..n] : s'_i = \tau(s_i, +m_i)$ and $c'_{i\ominus 1 i} = c''_{i\ominus 1 i}$.

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 sending cycle $\mathcal{C}_i$, respectively. The set of fair reachable states, denoted as $\mathbf{F}$, is called the *fair reachable state space* of P.

Example (Cont'd): Figure 2 shows the fair reachability graph for the protocol in Figure 1. In this case, $\mathbf{F} = \{S^0, S^1\}$, where $S^0 = (1, 1, 1, 1, \epsilon, \epsilon, \epsilon, \epsilon)$ is the initial state while $S^1 = (2, 2, 1, 1, \epsilon, \epsilon, \epsilon, \epsilon)$ is another fair reachable state. The fair progress vectors in S^0 and S^1 are $(-a, +a, \lambda, \lambda)$ and $(-b, +b, \lambda, \lambda)$, respectively. Note that this protocol is unbounded since both P_1 and P_3 have a reachable sending cycle. However, $\mathbf{F}$ is finite.

Note that $S^0 \in \mathbf{F}$ is a state with equal channel length of zero. Furthermore, any fair progress vector in S^0 maintains the equal channel length property in the resulting state. Using this argument inductively, we can conclude that $\mathbf{F}$ is included in the set of reachable states with equal channel length. In the following subsection, we show that the converse is also true. Denote $\mathbf{F}_k, k \geq 0$, as the set of fair reachable states whose channel length is k. The following lemma is straightforward.

Lemma 4.1 Given a fair reachable state space $\mathbf{F}$, the following statments hold: (1) $\forall\, k, k', k \neq k' : \mathbf{F}_k \cap \mathbf{F}_{k'} = \emptyset$. (2) $\mathbf{F} = \bigcup_{k=0}^{\infty} \mathbf{F}_k$. (3) $\forall\, S \in \mathbf{F}_k$, if $S \mapsto_f S'$, then $S' \in \mathbf{F}_0 \cup \mathbf{F}_1$ when $k = 0$; $S' \in \mathbf{F}_{k-1} \cup \mathbf{F}_k \cup \mathbf{F}_{k+1}$ otherwise. (4) $\mathbf{F}_k$ is finite. In fact, $|\mathbf{F}_k| \leq (\prod_{i=1}^{n} |S_i|) * (\prod_{i=1}^{n} |M_{ii\oplus 1}|^k)$. (5) $\mathbf{F}$ is finite iff $\exists K : K \geq 0, \mathbf{F}_{K+1} = \emptyset$.

4.2 Partial Fair Execution Sequence

Let S and S' be two states such that $S \mapsto^* S'$. An *execution sequence* from S to S', denoted as e, is a sequence $X^0 \xrightarrow{\tau^1} X^1 \xrightarrow{\tau^2} \ldots \xrightarrow{\tau^k} X^k, k \geq 0$, such that $X^0 = S$, $\forall j \in [1..k] : X^{j-1} \mapsto X^j$ via transition τ^j, and $X^k = S'$. The length of e, denoted as $|e|$, is defined as the number of transitions in e, i.e., $|e| = k \geq 0$. The corresponding *local execution sequence* in P_i is denoted as e_i. The length of e_i, denoted as $|e_i|$, is defined as the number of transitions in e_i. We use the notation $e \triangleq \{e_1, e_2, \ldots, e_n\}$ to denote the correspondence among an execution sequence and its local execution sequences. $\{e_1, e_2, \ldots, e_n\}$ is called a *local execution sequence set* from S to S'. When $S = S^0$, $\{e_1, e_2, \ldots, e_n\}$ is called a local execution sequence set for S'. For each reachable state S, there exists at least one execution sequence. Let e and e' be two execution sequences for S, $e \equiv e'$ iff they have the same local execution sequence set. It is obvious that $\equiv$ is an equivalence relation over the set of execution sequences for S. Each local execution sequence set characterizes a set of execution sequences for S. For state exploration, it is sufficient to examine these local execution sequence sets for each reachable state.

Similarly, if S' is fair reachable from S, then there is a sequence $X^0 \xrightarrow{\vec{\mathbf{v}}_1} X^1 \xrightarrow{\vec{\mathbf{v}}_2} \cdots \xrightarrow{\vec{\mathbf{v}}_k} X^k, k \geq 0$, such that $X^0 = S$, $\forall j \in [1..k] : X^{j-1} \mapsto_f X^j$ via fair progress vector $\vec{\mathbf{v}}_j$, and $X^k = S'$. Such a sequence is called a *fair execution sequence* from S to S', denoted as fs. The length of fs, denoted as $|fs|$, is defined as the number of fair progress vectors in the sequence, i.e., $|fs| = k \geq 0$. We also use the notation $fs \triangleq \{e_1, e_2, \ldots, e_n\}$ to denote the correspondence among a fair execution sequence and its local execution sequences. $\{e_1, e_2, \ldots, e_n\}$ is called a *fair* local execution sequence set from S to S'. When $S = S^0$, $\{e_1, e_2, \ldots, e_n\}$ is called a fair local execution sequence set for S'. Note that if $S \in \mathbf{F}$, then $\forall j \in [0..k] : X^j \in \mathbf{F}$.

Given a reachable state S. Let $\{e_1, e_2, \ldots, e_n\}$ be a local execution sequence set for S. We construct a fair execution sequence $seq = X^0 \xrightarrow{\vec{\mathbf{v}}_1} X^1 \xrightarrow{\vec{\mathbf{v}}_2} \cdots \xrightarrow{\vec{\mathbf{v}}_k} X^k, k \geq 0$, such that $X^0 = S^0$, $\forall j \in [1..k] : X^{j-1} \mapsto_f X^j$ via $\vec{\mathbf{v}}_j$, and no fair progress vector can be derived from the remaining transitions from $\{e_1, e_2, \ldots, e_n\}$ in state X^k. It is not difficult to show that both seq and X^k are *unique* w.r.t $\{e_1, e_2, \ldots, e_n\}$. seq and X^k are called the *partial fair execution sequence* and the *fair precursor* for S w.r.t $\{e_1, e_2, \ldots, e_n\}$, respectively, denoted as pfs and $fp = (s_1^p, s_2^p, \ldots, s_n^p, c_{n1}^p, c_{12}^p, \ldots, c_{n-1n}^p)$.

Lemma 4.2 Let fp be the fair precursor for a reachable state S w.r.t $\{e_1, e_2, \ldots, e_n\}$. If $S \notin \mathbf{F}$, then $fp \neq S$ and 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 $|e_k| \neq 0$, then τ_k^p, the transition from e_k at s_k^p, is executable. (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 4.1 $\mathbf{F}$ is exactly the set of reachable states with equal channel length.

An important implication of this theorem is that the notion of fair reachability is *consistent* with the notion of fair execution sequence in the sense stated in the following theorem.

Theorem 4.2 Let $\{e_1, e_2, \ldots, e_n\}$ be a local execution sequence set for S. If $\{e_1, e_2, \ldots, e_n\}$ is a fair local execution sequence for S, then any other local execution sequence set $\{e_1', e_2', \ldots, e_n'\}$ for S is also a fair execution sequence for S. In other words, if S is fair reachable, then it is fair reachable via any execution sequence for S.

4.3 Finiteness of F

Given a cyclic protocol P. We perform fair reachability analysis for P by generating the fair reachable state space $\mathbf{F}$. In order for the procedure to terminate, $\mathbf{F}$ has to be finite. For $n = 2$, a sufficient condition has been established for P to have a finite $\mathbf{F}$, namely, one of the channels being bounded [4]. However, no necessary and sufficient condition, even for $n = 2$, has been established so far. On the other hand, it is known that $\mathbf{F}$ can be finite even if P has a reachable sending cycle. This motivates us to look for other factors for causing $\mathbf{F}$ to become infinite.

We first investigate the class of cyclic protocols without reachable sending cycles. We notice that for a cyclic protocol without sending cycles, the notion of unboundedness is equivalent to that of "simultaneous unboundedness".

Definition 4.1 A cyclic protocol P is *simultaneously unbounded* if for any constant $K \geq 0$, there exists a reachable state $S = (s_1, s_2, \ldots, s_n, c_{n1}, c_{12}, \ldots, c_{n-1n})$ such that $\forall\, i \in [1..n] : |c_{ii\oplus 1}| > K$; otherwise, it is *not* simultaneously unbounded.

Lemma 4.3 Given a cyclic protocol P without reachable sending cycles. If P is unbounded, then P is simultaneously unbounded.

Then, we show that for a simultaneously unbounded cyclic protocol, we can find a fair reachable state whose channels are simultaneously unbounded.

Lemma 4.4 Given a cyclic protocol $P = (P_1, P_2, \ldots, P_n)$, 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 $\forall\, i \in [1..n] : |c'_{ii\oplus 1}| \geq K$.

With these two lemmas, we can establish an equivalence between the finiteness of $\mathbf{R}$ and finiteness of $\mathbf{F}$ for the class of cyclic protocols without sending cycles.

Theorem 4.3 Given a cyclic protocol P without reachable sending cycles. $\mathbf{F}$ is finite iff $\mathbf{R}$ is finite.

A rephrase of this theorem gives us a necessary and sufficient condition for a cyclic protocol with a finite $\mathbf{F}$ to be unbounded, a generalization of the result in [5] for $n = 2$ to $n \geq 2$.

Theorem 4.4 Given a cyclic protocol P with a finite $\mathbf{F}$. P is unbounded iff it has a reachable sending cycle.

Now we can confirm that simultaneous channel unboundedness is the fundamental factor in causing $\mathbf{F}$ to become infinite.

Theorem 4.5 Given a cyclic protocol P. $\mathbf{F}$ is finite iff P is not simultaneously unbounded.

This necessary and sufficient condition provides an exact description of the class of cyclic protocols with finite fair reachable state spaces from the protocol operational semantics viewpoint. To the best of our knowledge, this is the first necessary and sufficient condition for a cyclic protocol to have a finite fair reachable state space. However, as expected, the decidability aspect of this condition is negative, as is stated in the following theorem. The proof of the theorem is based on showing it is true for $n = 2$, an easy reduction by using the decidability result of boundedness detection established in [5].

Theorem 4.6 It is undecidable whether a cyclic protocol P has a finite $\mathbf{F}$.

The next theorem says that if a cyclic protocol has a finite $\mathbf{F}$, then we will be able to find the least upper bound $K \geq 0$ such that each reachable state has at least one channel whose length is bounded by K.

Theorem 4.7 Given a cyclic protocol P with a finite $\mathbf{F}$. Let K be the longest channel length among all the states in $\mathbf{F}$. Then each reachable state of P has at least one channel whose length is bounded by K.

Denote $\mathcal{P}$ as the class of cyclic protocols whose $\mathbf{F}$'s are finite. From now on, we will restrict our study to class $\mathcal{P}$. In the rest of the paper, unless otherwise stated explicitly, when we mention a cyclic protocol P, we mean $P \in \mathcal{P}$; when we mention $\mathbf{F}$, we mean that it is finite.

4.4 Fault Coverage of F

As with states in $\mathbf{R}$, we define logical errors for states in $\mathbf{F}$ in a similar way. Give a state $S \in \mathbf{F}$. S is a *fair* deadlock state iff S is a receiving state and all the channels are empty. S is a *fair* unspecified reception state iff there is a receiving local state $s_i \in S$ such that (i) $c_{i\ominus 1 i} = m \cdot c'_{i\ominus 1 i}$ and $\tau(s_i, +m)$ is not defined, or (ii) $c_{i\ominus 1 i} = \epsilon$, $\tau(s_{i\ominus 1}, -m)$ is defined, and $\tau(s_i, +m)$ is not defined.

Several comments are in order here. A fair unspecified reception state S is not an unspecified reception state when $c_{i\ominus 1 i} = \epsilon$. However, let S' be the following state by executing transition $\tau(s_{i\ominus 1}, -m)$ in S, S' will be an unspecified reception state. In other words, in fair reachability analysis, unspecified reception is sometimes detected in a "look-ahead" manner due to the incorporation of enabled transitions in fair progress vectors. Second, there might be "dead end" states in $\mathbf{F}$ whose $V = \emptyset$. However, it is not difficult to show that in this case S is either a fair deadlock state or a fair unspecified reception state. Thus the occurrence of dead end states does not introduce new types of logical errors in $\mathbf{F}$. Third, for unboundedness detection, we know from Theorem 4.4 that we only need to detect reachable sending cycles. As a result, we define a reachable state S as an *unbounded* state iff it is in a sending cycle. S is a *fair* unbounded state iff it is an unbounded state in $\mathbf{F}$.

State reduction achieved by fair reachability analysis is measured by the factor $\mathbf{R} \setminus \mathbf{F}$. In general, $\mathbf{F}$ is much smaller than $\mathbf{R}$, thus the saving is substantial. However, the study of logical error coverage of $\mathbf{F}$ is crucial to evaluating the usefulness of fair reachability analysis. For $n = 2$, it was shown that both deadlock and unspecified reception are detectable within $\mathbf{F}$ [16], and that unboundedness can be detected via finite extension of $\mathbf{F}$ [5]. Note that even for $n = 2$, nonexecutable transition detection has not been studied in the context of $\mathbf{F}$. For $n \geq 2$, we know that $\mathbf{F}$ is exactly the set of reachable states with equal channel length. Since all deadlock states are of equal channel length zero, we have the following theorem on deadlock detection.

Theorem 4.8 Deadlock detection is decidable for $\mathcal{P}$.

In fact, we showed in [8] that *livelock* detection is also decidable for $\mathcal{P}$. However, it is not difficult to see that for detection of logical errors other than deadlock, $\mathbf{F}$ is not sufficient, and thus finite extension of $\mathbf{F}$ is needed.

Example (Cont'd): In Figure 2, both S^0 and S^1 are fair unbounded states. However, just inspecting $\mathbf{F}$ cannot detect unboundedness caused by P_3 and an unspecified reception in channel

C_{34}. In this case, the behavior of P_3 and P_4 were not explored during state generation. As we will see in the following section, this is caused by the sending cycle in P_1, and this is not a coincidence.

Following the same formulation as [5], we reduce the detection of logical errors other than deadlock in $\mathcal{P}$ 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 $m \in M_{i \ominus 1 i}$, decide whether (m, s_i) is reachable.*

It should be clear that for $\mathcal{P}$, if we can solve **P-I** (**P-II**), then we can solve unboundedness (unspecified reception) detection, and 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 [1], we will show that both of them are decidable for $\mathcal{P}$ in the following section.

5 Finite Extension of F

In this section, we study the finite extension of **F** to detect logical errors other than deadlock for $\mathcal{P}$. For brevity, we use the term " logical errors" for "logical errors other than deadlock" in the rest of this section, unless otherwise explicitly stated.

Intuitively, there is a simple argument that shows that both **P-I** and **P-II**, and thus all logical errors, are decidable for the class of cyclic protocols $\mathcal{P}$ via fair reachability plus finite extension. Then the only issue remaining is how efficient the process is. The argument goes as follows: If a local state s_k is reachable, then there is a reachable state X such that $s_k \in X$. Thus, there is a local execution sequence set $\{e_1, e_2, \ldots, e_n\}$ from S^0 to X in **R**. From $\{e_1, e_2, \ldots, e_n\}$, a partial fair execution sequence pfs can be derived to get to fp. Note that fp is in the path of the execution sequence from S^0 to X and $fp \in$ **F**. If $s_k \in fp$, then we are done. Otherwise, from fp, a finite extension in **R** exists – simply the remainder of the execution sequence from fp to X. Hence, s_k is locatable by finite extension from **F**. A similar argument can be used for the reachability of (m, s_k).

What are the problems with this argument? First, it only shows existence but no algorithm. Secondly, it provides no upper bound on how far to extend when the execution sequence in **R** is unknown (as is generally the case). Yet, the preceding argument can serve as a general guideline in understanding the formality presented below.

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. In what follows, we first identify a necessary condition for the existence of such local states. Then we show how to minimize the size of the *extension set*, i.e., the set of states in **F** that need to be extended. In Subsection 5.2, we present a finite extension procedure based on partial states from the extension set. In the last subsection, we summarize the discussions with the decidability results for **P-I** and **P-II**, and a characterization of **F** in terms of logical error coverage.

5.1 Identifying the Extension Set

Suppose s_k is reachable but not fair reachable. Then none of the reachable states containing s_k is in **F**. Let X be any reachable state with $s_k \in X$, and $\{e_1, e_2, \ldots, e_n\}$ be a local execution sequence set for X. Let pfs and fp be the partial fair execution sequence and the fair precursor for X w.r.t $\{e_1, e_2, \ldots, e_n\}$, respectively. Based on the preceding discussion, it is clear that $s_k^p \neq s_k$ and thus $|e_k| \neq 0$. Furthermore, we can find a maximal interval $[i..k]$ in fp such that

$\forall j \in [i..k] : |e_j| \neq 0$. Note that each τ_j^p from e_j at s_j^p is executable. When $i \neq k$, the execution of remaining transitions in e_k depends only on the execution of transitions in e_j, $j \in [i..k \ominus 1]$.

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 $(j \in [i..k]) \wedge (|e_j| \neq 0)$. Note that during the construction, some of the e_j's may become empty when $(j \in [i..k]) \wedge (i \neq k)$. However, not in any of these states can e_k become empty since s_k is not fair reachable. 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. Obviously, $\mathbf{F}^{min}_{[i..k]} \subseteq \mathbf{F}$, and is closed by the above construction, i.e., 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 S does not have a concurrency vector, there is no send-receive pair in $\vec{\mathbf{u}}_{[i..k]}$, and neither u_i nor u_k appears 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 5.1 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.

As a result, the finite extension of $\mathbf{F}$ should be based on those states in $\mathbf{F}$ whose $W_{[i..j]} \neq \emptyset$ for some interval $[i..j]$. However, there are two problems we need to consider. First, there could be many states in $\mathbf{F}$ whose $W_{[i..j]} \neq \emptyset$. Finitely extending all such states can be costly and might incur considerable redundant work. Secondly, testing whether $W_{[i..j]} \neq \emptyset$ in S can also be costly because it involves checking all the states fair reachable from S without progress in $[i..j]$. Thus, instead of computing all the states whose $W_{[i..j]} \neq \emptyset$, 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 $\mathbf{FRG}_{[i..j]}$ such that the set of the nodes in the graph corresponds to the set of states fair reachable from S without progress in $[i..j]$ and S is the initial node of the graph. Then we construct the *quotient* graph $\mathbf{QFRG}_{[i..j]}$ such that each node is a strongly connected component (SCC) in $\mathbf{FRG}_{[i..j]}$. From graph theory, this graph is a directed acyclic graph (DAG). Each node in $\mathbf{QFRG}_{[i..j]}$ is denoted as $[S']$, where S' is a state in that SCC. The initial node is denoted as $[S]$. Let TN be the set of terminal nodes in $\mathbf{QFRG}_{[i..j]}$. Observe that $W_{[i..j]}(S'') \subseteq W_{[i..j]}(S')$ if S'' is fair reachable from S' without progress in $[i..j]$. Thus, $W_{[i..j]}(S'') = W_{[i..j]}(S')$ if S' and S'' are in the same SCC of $\mathbf{QFRG}_{[i..j]}$. Hence, we can use the notation $W_{[i..j]}([S'])$ to represent the set of *ppitv*'s in any state in $[S']$. In addition, it is not difficult to show 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 $[S']$. In either 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 5.2 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 a fair unbounded state.

Let $\mathbf{F}_T = \mathbf{F}_{ur} \cup \mathbf{F}_{ub}$, where $\mathbf{F}_{ur}$ and $\mathbf{F}_{ub}$ are the set of unspecified reception states and unbounded states in $\mathbf{F}$, respectively. Clearly, the extension set $\mathbf{F}_T$ for $\mathbf{F}$ can be easily computed during the construction of $\mathbf{F}$. Furthermore, based on the preceding discussion, we have the following lemma:

Lemma 5.3 There is a state $S \in \mathbf{F}$ whose $W_{[i..j]} \neq \emptyset$ only if there is a state $S' \in \mathbf{F}_T$ whose $U_{[i..j]} \neq \emptyset$ and $W_{[i..j]}(S) \subseteq U_{[i..j]}(S')$.

Therefore, to solve both **P-I** and **P-II** for $\mathcal{P}$, we only need to extend those states in $\mathbf{F}_T$. Now the question becomes how states in $\mathbf{F}_T$ can be extended in a finite way.

5.2 Partial State Exploration

From the previous subsection, we know that in order to solve both **P-I** and **P-II**, we only need to extend parts of a state S indexed by an interval $[i..j]$ whenever $U_{[i..j]} \neq \emptyset$, for each $S \in \mathbf{F}_T$. Such an interval $[i..j]$ is called a *proper incompatible* interval in S. Denote IJ as the set of proper incompatible intervals in S. Clearly, $(IJ, \subseteq)$ is a partial order set. Let ImJ be the set of maximal elements in $(IJ, \subseteq)$. Each element in ImJ is called a *maximal* proper incompatible interval in S. As will be clear shortly, for finite extension on S, we only need to consider those intervals in ImJ.

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 $S \in \mathbf{F}_T$. Given a state S and an interval $[i..j]$. A *partial state* in S indexed by $[i..j]$ is the set of local states and input channel contents of the processes indexed by $[i..j]$ in S, denoted as $PS_{[i..j]} = (c_{i\ominus 1i}, s_i, \ldots, c_{j\ominus 1j}, s_j)$. $PS'_{[i..j]} \subseteq PS_{[i..j]}$ iff $([i'..j'] \subseteq [i..j]) \wedge (\forall k \in [i'..j'] : (s_k = s'_k) \wedge (c_{k\oplus 1} = c'_{k\oplus 1}))$. We also use $PS_{[i..j]} \subseteq S$ to denote $PS_{[i..j]}$ is a partial state of S.

The reachability relation among partial states is defined as follows. $PS_{[i..j]} \mapsto_{[i..j]} PS'_{[i..j]}$ iff $\exists k \in [i..j]$ such that one of the following three conditions holds: (1) $k = j$ and $\tau(s_j, -m)$ is defined. Then $s'_j = \tau(s_j, -m)$. (2) $k \neq j$ and $\tau(s_k, -m)$ is defined. Then $c'_{kk\oplus 1} = c_{kk\oplus 1} \cdot m$ and $s'_k = \tau(s_k, -m)$. (3) $c_{k\ominus 1k} = m \cdot c''_{k\ominus 1k}$ and $\tau(s_k, +m)$ is defined. Then $c'_{k\ominus 1k} = c''_{k\ominus 1k}$ and $s'_k = \tau(s_k, +m)$. Except for the changes made above, all other elements of $PS'_{[i..j]}$ remain the same as $PS_{[i..j]}$. The reflexive, transitive closure of $\mapsto_{[i..j]}$ is denoted as $\mapsto^*_{[i..j]}$. $PS'_{[i..j]}$ is *reachable* from $PS_{[i..j]}$ iff $PS_{[i..j]} \mapsto^*_{[i..j]} PS'_{[i..j]}$. We can also define the reachability from $PS_{[i..j]}$ for s_k, (m, s_k), and sending cycle $\mathcal{C}_k$, respectively.

The following lemma justifies that we only need to perform finite extension on those partial states indexed by maximal proper incompatible intervals for each state in $\mathbf{F}_T$. When we set $[i..j] = [1..n]$, it also explains why we can base our finite extension of $\mathbf{F}$ on reachability among partial states.

Lemma 5.4 Given two partial states $PS_{[i'..j']}$ and $PS'_{[i'..j']}$. Suppose $PS_{[i'..j']} \subseteq PS_{[i..j]}$. Then $PS_{[i'..j']} \mapsto^*_{[i'..j']} PS'_{[i'..j']}$ only if $\exists\, PS'_{[i..j]} : (PS'_{[i'..j']} \subseteq PS'_{[i..j]}) \wedge (PS_{[i..j]} \mapsto^*_{[i..j]} PS'_{[i..j]})$.

It should be noted that the set of partial states reachable from $PS_{[i..j]}$ can be infinite. Thus, care must be taken in order to obtain a finite extension of $PS_{[i..j]}$. Denote $PS_{[i..j]} = (c_{i\ominus 1i}, s_i, \ldots, c_{j\ominus 1j}, s_j)$. Let $K_k = |c_{k\ominus 1k}|, k \in [i..j]$. $PS'_{[i..j]}$ satisfies the *channel constraint* w.r.t

$PS_{[i..j]}$ iff $\forall k \in [i..j] : |c'_{k\ominus 1k}| \leq MAX(K_k, 1)$. $PS_{[i..j]} \mapsto_{m[i..j]} PS'_{[i..j]}$ iff $PS_{[i..j]} \mapsto_{[i..j]} PS'_{[i..j]}$ and $PS'_{[i..j]}$ satisfies the channel constraint w.r.t $PS_{[i..j]}$. Denote $\mapsto^*_{m[i..j]}$ as the transitive, reflexive closure of $\mapsto_{m[i..j]}$. $PS'_{[i..j]}$ is *m-reachable* from $PS_{[i..j]}$ iff $PS_{[i..j]} \mapsto^*_{m[i..j]} PS'_{[i..j]}$. We can also define the m-reachability from $PS_{[i..j]}$ for s_k, (m, s_k), and sending cycle $\mathcal{C}_k$, respectively. By induction on the number of transitions executed from $PS_{[i..j]}$ to $PS'_{[i..j]}$, it can be shown that $PS_{[i..j]} \mapsto^*_{m[i..j]} PS'_{[i..j]}$ only if $PS'_{[i..j]}$ satisfies the channel constraint w.r.t $PS_{[i..j]}$.

To obtain a finite extension of $PS_{[i..j]}$, we construct a m-reachability graph $\mathbf{MRG}_k$ for each $PS_{[i..k]}, k \in [i..j]$, where the set of nodes corresponds to the set of m-reachable partial states from $PS_{[i..k]}$ and $PS_{[i..k]}$ is the initial node of the graph. By the channel constraint of $PS_{[i..k]}$, it follows that each $\mathbf{MRG}_k$ is finite, and so is $\mathbf{MRG}_{[i..j]} = \bigcup_{k \in [i..j]} \mathbf{MRG}_k$. Keep in mind that we are to solve the local reachability problems **P-I** and **P-II**. With the finite extension graph $\mathbf{MRG}_k$ for $PS_{[i..k]}$, it should be clear that if s_k ((m, s_k)) is m-reachable from $PS_{[i..k]}$, then it is reachable from $PS_{[i..k]}$. The question is whether the converse is also true.

To answer this question, we adopt a technique called *maximal progress partial state exploration* similar to the ones proposed in [6, 11]. Suppose $PS_{[i..j]} \mapsto^*_{[i..j]} PS^d_{[i..j]}$ via local execution sequence set $\{e_i, e_{i\oplus 1}, \ldots, e_j\}$. For any $s^d_k \in PS^d_{[i..j]}, k \in [i..j]$, we want to construct a partial state $PS^b_{[i..k]} = (c^b_{i\ominus 1i}, s^b_i, \ldots, c^b_{k\ominus 1k}, s^b_k)$ such that $(s^d_k = s^b_k) \wedge (PS_{[i..k]} \mapsto^*_{m[i..k]} PS^b_{[i..k]})$ by rearranging the execution order of the transitions in $\{e_i, e_{i\oplus 1}, \ldots, e_k\}$. Specifically, we let P_k maximally progress along e_k to reach s^d_k. In the following procedure, $\tau'_l, l \in [i..k]$, is the transition at s'_l from e_l in $PS'_{[i..k]}$. Denote $\vec{\tau}'_{[i..k]} = (\tau'_i, \tau'_{i\ominus 1}, \ldots, \tau'_k)$. For brevity, we define $\mathbf{max}(\vec{\tau}'_{[i..k]}) = l$ if l is the first executable transition in the order from k to i from $\vec{\tau}'_{[i..k]}$ in $PS'_{[i..k]}$; $\mathbf{max}(\vec{\tau}'_{[i..k]}) = 0$ if no such transition can be found in $PS'_{[i..k]}$.

Algorithm 1 Maximal-Progress
begin
1 $PS'_{[i..k]} := PS_{[i..k]}$
2 **while** $s'_k \neq s^d_k$ **do**
3 $\quad l := \mathbf{max}(\vec{\tau}'_{[i..k]})$
4 $\quad$ **let** $PS''_{[i..k]} :=$ apply τ'_l in $PS'_{[i..k]}$
5 $\quad PS'_{[i..k]} := PS''_{[i..k]}$
6 **end while**
7 $PS^b_{[i..k]} := PS'_{[i..k]}$
8 **return** $PS^b_{[i..k]}$
end Maximal-Progress

The correctness of the algorithm can be argued informally as follows. At each iteration from Line 2 to Line 6, if $s'_k \neq s^d_k$, then $\vec{\tau}'_{[i..k]}$ has at least one executable transition in $PS'_{[i..k]}$. After the execution of one iteration, the total number of transitions remained in $\{e_i, e_{i\oplus 1}, \ldots, e_k\}$ is reduced by one. Therefore, the algorithm must terminate within finite number of iterations, and when it terminates, $s^b_k = s^d_k$ in $PS^b_{[i..k]}$. By induction on the number of iterations from line 2 to line 6, it is not difficult to show that $PS_{[i..k]} \mapsto^*_{m[i..k]} PS'_{[i..k]}$. Thus at the end of the algorithm, $PS_{[i..k]} \mapsto^*_{m[i..k]} PS^b_{[i..k]}$.

As for (m, s^d_k), we first use the above algorithm to get to $PS^b_{[i..k]}$. If $(m, s^d_k) \in PS^b_{[i..k]}$, then we are done. Otherwise, since there is no transition left in e_k in $PS^b_{[i..k]}$, we must have $i \neq k$, $c^b_{k\ominus 1k} = \epsilon$, and the first sending transition in the remaining transitions in $e_{k\ominus 1}$ must be $\tau(s'_{k\ominus 1}, -m)$ for some local state $s'_{k\ominus 1}$ in $P_{k\ominus 1}$. Let $s''_{k\ominus 1} = \tau(s'_{k\ominus 1}, -m)$. Then we apply the above algorithm from $PS^b_{[i..k\ominus 1]}$ based on the remaining transitions in $\{e_i, e_{i\oplus 1}, \ldots, e_{k\ominus 1}\}$ to get to $PS^c_{[i..k\ominus 1]}$ such that $PS^b_{[i..k\ominus 1]} \mapsto^*_{m[i..k\ominus 1]} PS^c_{[i..k\ominus 1]}$ and $s^c_{k\ominus 1} = s''_{k\ominus 1}$. Let $c^c_{k\ominus 1k} = m$ and $s^c_k = s^b_k$.

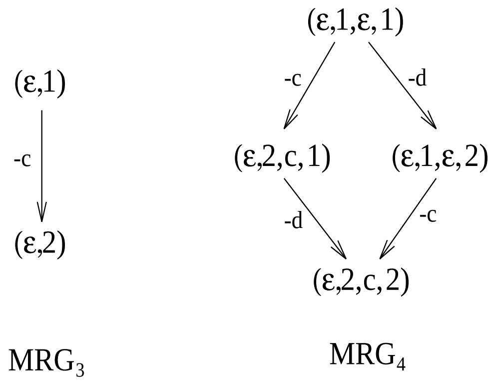

Figure 3: Finite Extension of Paritial State

Then $PS^b_{[i..k]} \mapsto^*_{m[i..k]} PS^c_{[i..k]}$ and $(m, s^d_k) \in PS^c_{[i..k]}$. Thus $PS_{[i..k]} \mapsto^*_{m[i..k]} PS^c_{[i..k]}$. To summarize, we have the following result:

Theorem 5.1 Given a partial state $PS_{[i..j]}$. Let $k \in [i..j]$. s^d_k is reachable from $PS_{[i..j]}$ iff it is m-reachable from $PS_{[i..k]}$. (m, s^d_k) is reachable from $PS_{[i..j]}$ iff it is m-reachable from $PS_{[i..k]}$. Therefore, both **P-I** and **P-II** are decidable for any partial state via finite extension on $PS_{[i..j]}$.

Example (Cont'd): In Figure 2, we have $\mathbf{F}_T = \mathbf{F}_{ub} = \mathbf{F}$. It should be clear that there is only one *ppitv* in the protocol, namely $\vec{\mathbf{u}}_{[3..4]} = (-c, -d)$. Hence, only one partial state $PS_{[3..4]} = (\epsilon, 1, \epsilon, 1)$ needs to be extended. The partial state m-reachability graph is shown in Figure 3, in which the "hidden local states" 2 in P_3 and $(c, 2)$ in P_4 are uncovered.
End of Example.

5.3 Fault Coverage of F Revisited

Let's recapitulate the discussion so far on finite extension of **F**. We began by observing that **F** itself is not sufficient for detection of logical errors, and then reduced the logical error detection problems to two local reachability problems **P-I** and **P-II**. We found that the major obstacle to solving these two problems is the existence of *ppitv*'s in some states in **F** since these *ppitv*'s prevent the fair progress state exploration procedure from reaching some reachable local states. However, we noticed that it suffices to do finite extension in those states in $\mathbf{F}_T$ in order to uncover all the "hidden" reachable local states. We further observed that only those partial states in each $S \in \mathbf{F}_T$ indexed by some interval in ImJ of S are needed for extension in order to solve both **P-I** and **P-II** for $\mathcal{P}$. Finally, we showed that both problems are solvable for any partial state via finite extension. Hence, we are able to establish the following decidability result:

Theorem 5.2 Both **P-I** and **P-II** are decidable for $\mathcal{P}$. Therefore, detection of unspecified reception, unboundedness and nonexecutable transition are all decidable for $\mathcal{P}$.

During the process, we have also discovered the following important characterization of **F** in terms of fault coverage.

Theorem 5.3 Given a cyclic protocol $P \in \mathcal{P}$. P has an unspecified reception but $\mathbf{F}_{ur} = \emptyset$ only if $\mathbf{F}_{ub} \neq \emptyset$. P is unbounded but $\mathbf{F}_{ub} = \emptyset$ only if $\mathbf{F}_{ur} \neq \emptyset$. P has a nonexecutable transition that is not detectable via $\mathbf{F}$ only if $\mathbf{F}_{ur} \cup \mathbf{F}_{ub} \neq \emptyset$. Therefore, P is logically correct iff $\mathbf{F}$ does not contain any logical errors.

Combined with the result that deadlock detection is decidable via $\mathbf{F}$ in Subsection 4.4, we can see that $\mathbf{F}$ is very competitive in fault coverage, quite to the contrary of the pessimistic suspicion from the surface at the beginning of this section. For iterative validation, we may stop state exploration whenever an error state is found in $\mathbf{F}$, fix that error, and repeat the process until no more errors are found in $\mathbf{F}$. In this way, finite extension of $\mathbf{F}$ is not necessary. If we want to detect all the errors in one–phase generation of $\mathbf{F}$, then $\mathbf{F}$ might need to be finitely extended if $\mathbf{F}_T \neq \emptyset$. In this case, the finite extension procedure can be optimized for efficiency. We have already seen how time complexity can be reduced by limiting finite extension to only those states in $\mathbf{F}_T$, and for each such state, to only those partial states indexed by intervals from ImJ of that state. In fact, we can do better by fine-tuning the decision procedures. For example, if we are to detect unspecified reception, finite extension is necessary only when $\mathbf{F}_{ur} = \emptyset$ and $\mathbf{F}_{ub} \neq \emptyset$, and is only performed on those states in $\mathbf{F}_{ub}$. Detection of other errors can be fine-tuned in a similar way. As for space complexity, $\mathbf{F}$ itself already offers substantial reduction of $\mathbf{R}$. When finite extension is needed, the additional space for an extension graph $\mathbf{MRG}_k$ is usually small compared to the size of $\mathbf{F}$ due to the channel constraint. Moreover, after $\mathbf{MRG}_k$ is generated, we can check logical errors in $\mathbf{MRG}_k$, mark the corresponding state accordingly if there is an error, and then discard it for good. Using this strategy, considerable space can be saved, especially when n gets larger. This is in contrast to the approach taken in [5] for $n = 2$, which keeps all the extension graphs during unboundedness detection. In summary, for the class of cyclic protocols $\mathcal{P}$, generalized fair reachability analysis has very competitive error-detection capability, and can be carried out both effectively and efficiently.

6 Conclusion

In this paper, we generalized the fair reachability analysis technique to cyclic protocols with $n \geq 2$ communicating finite state machines. Given a cyclic protocol P, we showed that its fair reachable state space $\mathbf{F}$ is exactly the set of reachable states with equal channel length, and established the lack of simultaneous unboundedness in P as a necessary and sufficient condition for $P \in \mathcal{P}$, the class of cyclic protocols whose $\mathbf{F}$'s are finite. The effectiveness of generalized fair reachability analysis was demonstrated by showing for class $\mathcal{P}$, deadlock is detectable within $\mathbf{F}$ while all other logical errors are detectable via finite extension of $\mathbf{F}$. More importantly, we discovered a characterization of $\mathbf{F}$ in terms of fault coverage, namely P is logically correct iff $\mathbf{F}$ is free of logical errors.

Fair reachability analysis was originally proposed as a technique to tackle state explosion during reachability analysis [16]. The same argument also applies to our work reported in this paper. By forcing the system to progress through a fair execution sequence, we have cut down the redundancy of state exploration due to equivalent execution sequences. We also showed how finite extension can be carried out efficiently in terms of both time and space by minimizing the extension set and the number of partial states needed to be extended for each state in the extension set.

The strength of our approach lies in the natural generalization of the existing fair reachability technique and its simple, straightforward, and efficient decision procedures, which were missing in both [14, 15] and [11, 12]. This study shows that generalized fair reachability analysis not only achieves substantial state reduction, but also maintains very competitive logical error detection capability. Therefore, it is a very useful technique to prove logical correctness for a wide variety of cyclic protocols.

During the write-up of this paper, we were informed of the independent work by Peng on extending fair reachability to a model called "single-link communicating finite state machines" [13]. In this model, each process can have multiple output channels but has only one common input channel to store messages from other processes. Although cyclic protocols are included in this model, the notion of fair reachability in this model is quite different from ours in that only two machines are allowed to make progress at one time restricted by the so-called "weight-balance" constraint in [13]. It is not clear, however, what class of protocols in his model is amendable for his analysis technique. For cyclic protocols, our fair reachability formulation has the following advantages: First, our fair reachability state space maintains the same nice equal channel length property as for $n = 2$ [16, 5]. Second, both concurrency and synchronization vectors in our fair reachability notion allow more than two machines to progress at the same time. As a result, for most cyclic protocols, our analysis achieves greater state reduction than the one in [13]. Third, aside from deadlock, our approach can also detect other logical errors such as unspecified reception, nonexecutable transition, and unboundedness, which are not covered in [13].

Many open problems remain concerning our approach. First, although we have found a necessary and sufficient condition for the class of cyclic protocols whose logical correctness is decidable, we are not sure how general it is in terms of tightening the boundary of cyclic protocols whose logical correctness is decidable. Further investigation of this aspect is necessary in order to fully evaluate its role in the decidability hierarchy. Second, a cyclic protocol is still simple in topology. It would be beneficial to look into the possibility of generalizing our work to protocols with more complicated and yet regular network topologies. Third, it is possible to incorporate internal transitions into the fair progress vector formulation to allow our technique to handle cyclic protocols with internal transitions and still achieve good state reduction. We are currently working on this issue. Fourth, fair reachability analysis is only one type of improved reachability analysis techniques studied in the two machine case. In this paper, the collective power of both fair progress and maximal progress state exploration is illustrated in the finite extension process, and has produced encouraging results. The results reported here should encourage more research on extending other existing techniques to the analysis of protocols with more than two machines. Finally, it would be interesting to investigate the possibility of carrying the fair reachability analysis technique over to other specification models, such as the extended finite state machine 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] T.Y. Choi and R.E. Miller, "Protocol Analysis and Synthesis by Structured Partitions," Computer Networks and ISDN Systems, Vol. 11, 1986, pp. 367–381.

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

[6] 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.

[7] H. Liu and R.E. Miller, "Deadlock Detection for Cyclic Protocols Using Generalized Fair Reachability Analysis," Technical Report CS-TR-3135, Dept. of Computer Science, Univ. of Maryland at College Park, September 1993.

[8] 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. 271-286.

[9] 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.

[10] K. Okumura, "Protocol Analysis from Language Structure," PSTV'88, S. Aggarwal and K. Sabnani (Ed.), 1988, pp. 113–124.

[11] J. Pachl, "Reachability Problems for Communicating Finite State Machines," Research Report, CS-82-12, Dept. of Computer Science, Univ. of Waterloo, May, 1982

[12] J. Pachl, "Protocol Description and Analysis Based on a State Transition Model with Channel Expressions," PSTV'87, J. Rubin and C.H. West (Ed.), 1987, pp. 207–219.

[13] W. Peng, "Single-Link and Time Communicating Finite State Machines," ICNP'94, Boston, Massachusetts, October 25–28, pp. 126–133.

[14] W. Peng and S. Purushothaman, "A Unified Approach to the Deadlock Detection Problem in Networks of Communicating Finite State Machines," CAV'90, New Brunswick, N.J., June, 1990, E.M. Clarke and R.P.Kurshan (Ed.), LNCS Vol. 531, pp. 243–252.

[15] W. Peng and S. Purushothaman, "Data Flow Analysis of Communicating Finite State Machines," ACM Transactions on Programming Languages and Systems, Vol. 13, No. 3, 1991, pp. 399–442.

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

[17] 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 4.2 Let fp be the fair precursor for a reachable state S w.r.t $\{e_1, e_2, \ldots, e_n\}$. If $S \notin \mathbf{F}$, then $fp \neq S$ and 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 $|e_k| \neq 0$, then τ_k^p, the transition from e_k at s_k^p, is executable. (4) $fp \mapsto^* S$ via the remaining transitions from $\{e_1, e_2, \ldots, e_n\}$ in fp.

Proof: Obviously, $fp \neq S$, otherwise S will be fair reachable. Since $\{e_1, e_2, \ldots, e_n\}$ is a local execution sequence set for S, 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 $|e_k| \neq 0$. 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 $c_{k \ominus 1 k}^p = \epsilon$ since 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 4.1 **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, 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 4.2, such an interval exists and $\forall\, k \in [i..j] : \tau_k^p$ is executable. 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 $C_{jj\oplus 1}$ cannot be decreased. By the time the protocol gets to S, the length of channel $C_{jj\oplus 1}$ will be 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 4.3 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 $e \triangleq \{e_1, e_2, \ldots, e_n\}$ 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 $e \triangleq \{e_1, e_2, \ldots, e_n\}$ 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 $C_{i i \oplus 1}$. 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 the length of each channel greater than K, i.e., P is simultaneously unbounded. ∎

Lemma 4.4 Given a cyclic protocol P. If there is a reachable state $S = (s_1, s_2, \ldots, s_n, c_{n1}, c_{12}, \ldots, c_{n-1n})$ such that $\forall\, i \in [1..n] : |c_{i i \oplus 1}| \geq K$ 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 $\forall\, i \in [1..n] : |c'_{i i \oplus 1}| \geq K$.

Proof: 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 $e \triangleq \{e_1, e_2, \ldots, e_n\}$ 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 4.1. Suppose fp is of channel length K'. If $K' \geq K$, then let $S' = fp$, and we are done. Suppose not, by Lemma 4.2, $\exists k \in [1..n] : |e_k| = 0$. Note that from state fp and on, the length of channel $C_{k k \oplus 1}$ 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 $C_{k k \oplus 1}$ will be less than K, which contradicts the fact that every channel in S has length no less than K. Hence, fp must have channel length no less than K.

In all cases, we can find a fair reachable state whose channel length is no less than K. ∎

Theorem 4.3 Given a cyclic protocol P without reachable sending cycles. $\mathbf{F}$ is finite iff $\mathbf{R}$ is finite.

Proof: $\mathbf{R}$ being finite implies that $\mathbf{F}$ is finite since $\mathbf{F} \subseteq \mathbf{R}$. Suppose $\mathbf{F}$ is finite but $\mathbf{R}$ is infinite, then P is unbounded. By Lemma 4.3, P is simultaneously unbounded. By Lemma 4.4, for any $K \geq 0$, we can find a fair reachable state whose channel length is greater than K. On the other hand, $\mathbf{F}$ being finite implies that the channel length of any fair reachable state is bounded by some constant $K' \geq 0$. Choosing any $K > K'$ will result in a contradiction. ∎

Theorem 4.4 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 4.3, P is simultaneously unbounded. By Lemma 4.4, there is a fair reachable state S whose channel length is greater than K for any $K \geq 0$. Hence, $\mathbf{F}$ is infinite for P. A contradiction. ∎

Theorem 4.5 Given a cyclic protocol P. $\mathbf{F}$ is finite iff P is not simultaneously unbounded.

Proof: *If Part.* Suppose $\mathbf{F}$ is infinite, then $\bigcup_{k=0}^{\infty} \mathbf{F}_k$ 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.

Only If Part. If P is simultaneously unbounded, then for any $K \geq 0$, there is a reachable state S such that the length of each channel in S is greater than K. By Lemma 4.4, there is a fair reachable state S' such that each channel length in S' is greater than K. In other words, $\forall K \geq 0 \exists K' > K : \mathbf{F}_{K'} \neq \emptyset$. As a result, $\mathbf{F} = \bigcup_{k=0}^{\infty} \mathbf{F}_k$ is infinite. ∎

Theorem 4.6 It is undecidable whether a cyclic protocol P has a finite $\mathbf{F}$.

Proof: We claim that for $n = 2$, it is undecidable whether a (cyclic) protocol P has a finite $\mathbf{F}$. We prove this claim by contradiction. In the proof, we make use of the decidability of boundedness detection for protocols with finite $\mathbf{F}$'s for $n = 2$, a result established in [5].

Suppose for $n = 2$, it is decidable whether a protocol has a finite $\mathbf{F}$. Then the following algorithm will decide whether P is bounded:

Step 1: Check if $\mathbf{F}$ is finite for P.
Step 2: If $\mathbf{F}$ is infinite, output "P is unbounded".
Step 3: If $\mathbf{F}$ is finite, determine if P is bounded and output the result.
Step 4: End of procedure.

On the other hand, we know that boundedness detection is undecidable for protocols with $n = 2$ machines [1]. A contradiction.

Now that it is undecidable, for $n = 2$, whether a cyclic protocol has a finite $\mathbf{F}$, it is straightforward that the theorem holds for $n \geq 2$. ∎

Theorem 4.7 Given a cyclic protocol P with a finite $\mathbf{F}$. Let K be the longest channel length among all the states in $\mathbf{F}$. Then each reachable state of P has at least one channel whose length is bounded by K.

Proof: Suppose there is a reachable state S in which each of its channel length is greater than K. Then by Lemma 4.4, there is a fair reachable state whose channel length is greater than K, which contradicts the selection of K. ∎

Lemma 5.2 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 a fair unbounded 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 $\forall\, k \in \overline{[i..j]} : E_k \cup E_k^{++} \neq \emptyset$. 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 $\{e_1', e_2', \ldots, e_n'\}$ be the corresponding local execution sequence set from S to S. Then $\forall\, k \in [i..j] : |e_k'| = 0$. Since $|fc| \neq 0$, there must be a nonempty interval $[h..l]$ in S such that $\{i..j\} \cap \{h..l\} = \emptyset$, $\forall\, k \in [h..l] : |e_k'| \neq 0$, and $|e_{h\ominus 1}'| = |e_{l\oplus 1}'| = 0$. Note that $\forall\, k \in [h..l] : e_k'$ is a cycle (not necessarily elementary) in the state transition graph of P_k. We claim that e_h' is a sending cycle in P_h. Suppose not. Then there is at least one receiving transition in e_h'. Assume S is of channel length K. Then going through fc once will decrease the length of channel $C_{h\ominus 1 h}$ 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, e_h' must be a sending cycle in P_h. As a result, S is a fair unbounded state. ∎

Lemma 5.4 Given two partial states $PS_{[i'..j']}$ and $PS'_{[i'..j']}$. Suppose $PS_{[i'..j']} \subseteq PS_{[i..j]}$. Then $PS_{[i'..j']} \mapsto^*_{[i'..j']} PS'_{[i'..j']}$ only if $\exists\, PS'_{[i..j]} : (PS'_{[i'..j']} \subseteq PS'_{[i..j]}) \wedge (PS_{[i..j]} \mapsto^*_{[i..j]} PS'_{[i..j]})$.

Proof: The lemma is trivially true if $[i'..j'] = [i..j]$. Suppose $[i'..j'] \subset [i..j]$. Let $\{e_{i'}, e_{i'\oplus 1}, \ldots, e_{j'}\}$ be a local execution sequence set from $PS_{[i'..j']}$ to $PS'_{[i'..j']}$. Let *seq* be the sequence of messages sent by the (sending) transitions in $e_{j'}$. Construct $PS'_{[i..j]}$ such that (1) $PS'_{[i'..j']} \subseteq PS'_{[i..j]}$, (2) $\forall\, k \in (\{i..j\} \backslash \{i'..j'\}) : s_k' = s_k$, and (3) $\forall\, k \in (\{i..j\} \backslash \{i'..j'\}) : c_{k\ominus 1 k} = seq$ if $k = j' \oplus 1$; $c_{k\ominus 1 k} = \epsilon$ otherwise. Then obviously, $PS_{[i..j]} \mapsto^*_{[i..j]} PS'_{[i..j]}$. ∎

Theorem 5.2 Both **P-I** and **P-II** are decidable for $\mathcal{P}$. Therefore, detection of unspecified reception, unboundedness and nonexecutable transition are all decidable for $\mathcal{P}$.

Proof: We first show **P-I** is decidable for $\mathcal{P}$. Let P be any protocol in $\mathcal{P}$ and s_k be a local state of $P_k, k \in [1.n]$. We want to decide if s_k is reachable. Clearly, it is decidable whether s_k is fair reachable by inspecting $\mathbf{F}$, the finite fair reachable state space of P. If s_k is fair reachable, then it is reachable.

Suppose s_k is not fair reachable. Then, by Lemma 5.1 and Lemma 5.3, s_k is reachable only if there is an $S \in \mathbf{F}_T$ such that $k \in [i'..j']$, $U_{[i'..j']} \neq \emptyset$, and s_k is reachable from S. More specifically, from the discussion in Subsection 5.1, s_k is reachable via partial state $PS_{[i'..j']} \subset S$. By Lemma 5.4, s_k is reachable from $PS_{[i..j]}$, where $[i..j]$ is a maximal proper incompatible interval in S and $[i'..j'] \subseteq [i..j]$. By Theorem 5.1, s_k is m-reachable from $PS_{[i..j]}$. Hence, if

s_k is reachable but not fair reachable, then it can be decided as reachable by checking each $\mathbf{MRG}_{[i..j]}$ for each maximal proper incompatible interval $[i..j]$ in each $S \in \mathbf{F}_T$. Since $\mathbf{F}_T$ is finite and $\mathbf{MRG}_{[i..j]}$ is finite for any $PS_{[i..j]}$, this can be done within finite number of steps.

On the other hand, if s_k is not reachable, then it is not fair reachable. Furthermore, it cannot be m-reachable from any $PS_{[i..j]}$ for any $S \in \mathbf{F}_T$, where $[i..j]$ is a maximal proper incompatible interval in S. Otherwise, by Theorem 5.1, s_k is reachable from $PS_{[i..j]}$ for some $S \in \mathbf{F}_T$. By Lemma 5.4, s_k is reachable from S. Hence, s_k becomes reachable. A contradiction. Therefore, if s_k is not reachable, it can also be decided as not reachable by first examining $\mathbf{F}$ and then checking each $\mathbf{MRG}_{[i..j]}$ for each maximal proper incompatible interval $[i..j]$ in each $S \in \mathbf{F}_T$. This also can be done within finite number of steps.

To sum up, the reachability of a local state s_k can be decided within finite number of steps. As a result, **P-I** is decidable for $\mathcal{P}$. The decidability of **P-II** for $\mathcal{P}$ can also be shown in a similar way. Now that both **P-I** and **P-II** are decidable for $\mathcal{P}$, it is straightforward that detection of unspecified reception, nonexecutable transition, and unboundedness are all decidable for $\mathcal{P}$. ∎

Theorem 5.3 Given a cyclic protocol $P \in \mathcal{P}$. P has an unspecified reception but $\mathbf{F}_{ur} = \emptyset$ only if $\mathbf{F}_{ub} \neq \emptyset$. P is unbounded but $\mathbf{F}_{ub} = \emptyset$ only if $\mathbf{F}_{ur} \neq \emptyset$. P has a nonexecutable transition that is not detectable via $\mathbf{F}$ only if $\mathbf{F}_{ur} \cup \mathbf{F}_{ub} \neq \emptyset$. Therefore, P is logically correct iff $\mathbf{F}$ does not contain any logical errors.

Proof: 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 5.1 and Lemma 5.3, $\mathbf{F}_T \neq \emptyset$. Since $\mathbf{F}_T = \mathbf{F}_{ur} \cup \mathbf{F}_{ub}$, we must have $\mathbf{F}_{ub} \neq \emptyset$.

The proofs for unboundedness 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 $\mathbf{F}$. P cannot have any other logical errors either since otherwise we will have $\mathbf{F}_T \neq \emptyset$ based on the discussion in the preceding paragraph. Hence, P is logically correct. ∎