Generalized Fair Reachability Analysis for Cyclic Protocols: Decidability for Logical Correctness Problems *

Hong Liu Raymond E. Miller

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

Abstract

In a previous paper, we generalized the fair reachability notion to cyclic protocols with $n \geq 2$ machines, and showed that deadlock detection is decidable for $\mathcal{P}$, the class of cyclic protocols whose fair reachable state spaces are finite. In this paper, we show that detection of unspecified receptions, unboundedness, and nonexecutable transitions are all decidable for class $\mathcal{P}$ via finite extension of the 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 viable state reduction technique.

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 insured, 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.

Fair reachability analysis was proposed as one of the improved reachability analysis techniques for protocols with two machines [9, 2]. 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 insured, although in some cases, finite extension of the reduced state space is necessary [2]. In [4, 5], we generalized the fair reachability notion to cyclic protocols with more than two machines. For the class of cyclic protocols whose fair reachable state spaces are finite, we showed that deadlock detection is decidable. We also found a necessary and sufficient condition for a cyclic protocol to have a finite fair reachable state space [5].

In this paper, we show that for any cyclic protocol with finite fair reachable state space, detection of logical errors other than deadlock are all decidable via finite extension of the fair reachable state space. Furthermore, we have discovered a complete characterization for fair reachable state space in terms of logical error coverage, with which the overhead of finite extension is kept to the minimum. Therefore, for the class of cyclic protocols whose fair reachable state spaces are finite, detection of logical errors are all decidable, and can be carried out efficiently, as compared to the approaches in [7, 8]. The results reported from our study show that fair reachability analysis not only achieves good state reduction but also maintains competitive logical error coverage, and thus is a viable state reduction technique.

The rest of the paper is organized as follows. In Section 2, the communicating finite state machine model is introduced. In Section 3, we first describe the concept and formulation of the generalized fair reachability notion for cyclic protocols. Then, we look at the

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

0-8186-6685-4/94 $4.00 © 1994 IEEE

limitations of fair reachability analysis and reduce the logical error detection problems studied in this paper to two local state reachability problems. The main results of this paper are presented in Section 4 in which we show how finite extension can be performed on a finite fair reachable space so that logical errors can be detected both effectively and efficiently. We summarize the paper with open problems in Section 5.

Due to space limitations, lemmas and theorems presented in the paper are stated without proof. Please refer to the full paper [6] for details.

2 The CFSM Model

Notation: (1) We use $\cdot$ to denote concatenation. Given a set M. M^* denotes its the 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 $\{i..j\}$. $[i'..j'] \subseteq [i..j]$ if and only if (iff for short) $\{i'..j'\} \subseteq \{i..j\}$. 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 $\sigma = -m$ ($\sigma = +m$). s_i is a *sending* (*receiving*) local state iff all transitions defined in s_i are sending (receiving) transitions. We use the notation $\tau_i = \tau(s_i, \sigma)$ to give a name τ_i 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. Similarly, a transition cycle $\mathcal{C}_i$ in P_i is a cycle in the transition graph of P_i. It is a *sending* cycle in P_i iff all the transitions in $\mathcal{C}_i$ are sending transitions.

A protocol $P = (P_1, P_2, \ldots, P_n)$ is *cyclic* iff each P_i has only one input channel $C_{i \ominus 1 i}$ and only one output channel $C_{i i \oplus 1}$. From now on, we are dealing with cyclic protocols. For results established later in this paper, it should be clear that they apply to cyclic protocols only.

For 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 in a *sending* cycle iff there is a local state s_i that is in a sending cycle in 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 $c_{i i \oplus 1}' = c_{i i \oplus 1} \cdot m$, (2) $s_i' = \tau(s_i, +m)$ and $c_{i \ominus 1 i} = m \cdot c_{i \ominus 1 i}'$. 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*. For ease of discussion, 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. Suppose S' is reachable (from S), a local state s_i' is *reachable* (from S) if $s_i' \in S'$; (m, s_i') is *reachable* (from S) if $(m, s_i') \in S'$. 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).

The set of reachable states in P is denoted as $\mathbf{R}$, called the *reachable state space* of P. For protocol validation, we check states in $\mathbf{R}$ against *logical errors* such as deadlock, unspecified reception, nonexecutable transition, and unboundedness. (We assume reader's familiarity of these concepts. Please refer to [1].) This state exploration technique is called *reachability analysis*. P is *logically correct* iff $\mathbf{R}$ is free of logical errors. However, it was shown in [1] that even for $= 2$, 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.

3 Generalized Fair Reachability Analysis

Fair reachability analysis was first introduced as a state reduction technique for protocols with $n = 2$ machines [9, 2]. In [5], we generalized this technique to cyclic protocols with $n \geq 2$ machines and proved some new results. In this section, we briefly introduce the concept of generalized fair reachability and related results. For a complete treatment, please refer to [5]. For conciseness, we use "fair reachability" for "generalized fair reachability" from now on.

3.1 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 $E_i = E_i^- + E_i^+$, 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, a *pseudo transition vector* in S is a n-tuple $\vec{t} = (t_1, t_2, \ldots, t_n)$ such that $\forall i \in [1..n] : t_i \in E_i \cup E_i^{++} \cup \{\lambda\}$, where λ stands for a *null* transition in P_i. Let $TV = \{\vec{t} = (t_1, t_2, \ldots, t_n)\}$ 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 $\vec{t} \in TV$, we compute a pseudo transition vector $\vec{v} = (v_1, v_2, \ldots, v_n)$ from $\vec{t}$ according to one of the following three cases:

(1) $\vec{t} \in (\times_{k=1}^n E_k^-) \cup (\times_{k=1}^n E_k^+)$. In this case, $\vec{v} = \vec{t}$. Here, $\vec{v}$ 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 $\vec{t}$. 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, $\vec{v}$ retains all the send-receive pairs in $\vec{t}$. $\vec{v}$ is called a *synchronization vector* in S.

(3) Neither condition for Case 1 nor condition for Case 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.

For each $\vec{v}$ thus computed, $\vec{v}$ is a *fair progress vector* in S iff it is either a concurrency vector or a synchronization vector in S. Let V_c (V_s) be 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.

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 $\exists \vec{v} \in V(S)$ that leads the system from S to S'. There are three cases to consider:

(1) $\vec{v} \in V_s(S)$. 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) $\vec{v} \in V_c(S) \wedge (\forall i \in [1..n] : v_i = \tau(s_i, -m_i) \in E_i^-)$. The result of applying $\vec{v}$ 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) $\vec{v} \in V_c(S) \wedge (\forall i \in [1..n] : v_i = \tau(s_i, +m_i) \in E_i^+)$. Assume that before applying $\vec{v}$, $\forall i \in [1..n] : c_{i\ominus 1 i} = m_i \cdot c''_{i\ominus 1 i}$. The result of applying $\vec{v}$ 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 C_i, respectively.

Given a cyclic protocol P, the set of fair reachable states, denoted as $\mathbf{F}$, is called the *fair reachable state space* of P. It was shown in [5] that $\mathbf{F}$ is exactly the set of reachable state with equal channel length. Notice that $\mathbf{F}$ can be infinite. We are particularly interested in $\mathcal{P}$, the class of cyclic protocols whose $\mathbf{F}$'s are finite. It was shown that a cyclic protocol $P \in \mathcal{P}$ iff it is not simultaneously unbounded [5]. (P is *simultaneously unbounded* iff $\forall K \geq 0$, $\exists S \in \mathbf{R}$ such that the length of each channel in S is greater than K.) From now on, we 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.

3.2 Limitations

As with states in $\mathbf{R}$, we define logical errors for states in $\mathbf{F}$ in a similar way, although there are two subtleties regarding $\mathbf{F}$. First, unspecified receptions are sometimes detected in a "look-ahead" fashion 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 can be shown that the definition of unspecified reception in $\mathbf{F}$ is *consistent* with that in $\mathbf{R}$, and the occurrence of dead end states do not introduce new types of logical errors in $\mathbf{F}$. (Please refer to the full paper [6] for details.) Furthermore, for unboundedness in cyclic protocols, we generalize the result in [2] for $n = 2$ to $n \geq 2$. As a result, we can define S as an *unbounded* state iff it is in a reachable sending cycle.

Theorem 3.1 Given a protocol $P \in \mathcal{P}$. P is unbounded iff there is a reachable sending cycle.

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. During the study, we differentiate two cases: *pure* fair reachability analysis, which is solely based on $\mathbf{F}$; and *extended* fair reachability analysis, which is based on $\mathbf{F}$ and the *finite extension* of $\mathbf{F}$.

For $n = 2$, it was shown that both deadlock and unspecified reception are detectable within $\mathbf{F}$ [9], and that unboundedness can be detected via finite extension of $\mathbf{F}$ [2]. Note that even for $n = 2$, nonexecutable transition detection has not been studied in the context of $\mathbf{F}$.

For $n > 2$, it was shown that $\mathbf{F}$ has a complete coverage of deadlocks [5]. 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. In the remainder of this section, we study what properties are to be considered during the extension and defer the issue of finite extension to the next section. For brevity, we use the term "logical errors" for "logical errors other than deadlock" in the rest of this paper, unless otherwise explicitly stated.

Following the same formulation as [2], we reduce the detection of logical errors 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.

4 Finite Extension of F

This section presents the main results of this paper. We begin with the outline of the basic idea behind our approach. Then, we present a technique of partial state exploration from which finite extension of $\mathbf{F}$ becomes feasible. We then study the characterization of $\mathbf{F}$ in terms of fault coverage and discuss how $\mathbf{F}$ can be extended *efficiently*.

4.1 The Basic Idea

Let $S = (s_1, s_2, \ldots, s_n, c_{n1}, c_{12}, \ldots, c_{n-1n})$ be a reachable state, and e be an execution sequence that leads the system from S^0 to S. Let $\{e_1, e_2, \ldots, e_n\}$ be the corresponding local execution sequence set. From $\{e_1, e_2, \ldots, e_n\}$, we can construct a *partial fair execution sequence* for S, denoted as $pfs(S) = X^0 \xrightarrow{\vec{\mathbf{v}}_1} X^1 \xrightarrow{\vec{\mathbf{v}}_2} \ldots \xrightarrow{\vec{\mathbf{v}}_k} X^k$, 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, denoted as $fp(S) = (s_1^p, s_2^p \ldots, s_n^p, c_{n1}^p, c_{12}^p \ldots, c_{n-1n}^p)$. (Strictly speaking, both $pfs(S)$ and $fp(S)$ should be quantified with $\{e_1, e_2, \ldots, e_n\}$. However, the notations used here are sufficient for the following presentation.) It can be shown that both $pfs(S)$ and $fp(S)$ are *unique* with respect to (w.r.t for short) $\{e_1, e_2, \ldots, e_n\}$ and $fp(S) \neq S$ iff $S \notin \mathbf{F}$. In addition, when $S \notin \mathbf{F}$, the following statements are true in $fp(S)$: (1) $\exists k \in [1..n] : |e_k| = 0$. (2) $\exists k \in [1..n] : |e_k| \neq 0$. (3) If $|e_k| \neq 0$, then τ_k^p, the transition from e_k at s_k^p is executable. (4) $fp(S) \mapsto^* S$ via the remaining transitions in $\{e_1, e_2, \ldots, e_n\}$. For details, please refer to [5, 6].

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 S such that $s_k \in S$. Thus, there is a local execution sequence set

$\{e_1, e_2, \ldots, e_n\}$ from S^0 to S in $\mathbf{R}$. From $\{e_1, e_2, \ldots, e_n\}$, a partial fair execution sequence $pfs(S)$ can be derived to get to $fp(S)$. Note that $fp(S)$ is in the path of the execution sequence from S^0 to S and $fp(S) \in \mathbf{F}$. If $s_k \in fp(S)$, then we are done. Otherwise, from $fp(S)$, a finite extension in $\mathbf{R}$ exists – simply the remainder of the execution sequence from $fp(S)$ to S. Hence, s_k is locatable by finite extension from $\mathbf{F}$. 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 $\mathbf{R}$ 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 $\mathbf{F}$ 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 $\mathbf{F}$ that need to be extended. In Subsection 4.3, 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 $\mathbf{F}$ in terms of logical error coverage.

4.2 Identifying the Extension Set

Suppose s_k is reachable but not fair reachable. Then none of the reachable states containing s_k is in $\mathbf{F}$. 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 $pfs(S)$ and $fp(S)$ be the partial fair execution sequence and the fair precursor for S 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 $fp(S)$ 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 $fp(S)$, we construct the set of states fair reachable from $fp(S)$ 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 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'.

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, we can show $W_{[i..j]}(S) = \bigcap_{[S'] \in TN} W_{[i..j]}([S'])$. 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 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 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, 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 $U_{[i..j]} \neq \emptyset$. Therefore, to solve both **P-I** and **P-II** for $\mathcal{P}$, we only need to finitely extend those states in $\mathbf{F}_T$.

4.3 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. It can be shown [6] that for any $S \in \mathbf{F}_T$, ImJ can be computed both *effectively* and *efficiently*, and the elements in ImJ are mutually disjoint and not adjacent to each other. (Two intervals $[i..j]$ and $[i'..j']$ are *adjacent* iff $(i = j' \oplus 1) \vee (i' = j \oplus 1)$.)

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 *configuration* 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 $CF_{[i..j]} = (c_{i \ominus 1 i}, s_i, .., c_{j \ominus 1 j}, s_j)$. $CF'_{[i..j]} \subseteq CF_{[i..j]}$ 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 $CF_{[i..j]} \subseteq S$ to denote $CF_{[i..j]}$ is a configuration of S.

The reachability relation among configurations is defined as follows. $CF_{[i..j]} \mapsto_{[i..j]} CF'_{[i..j]}$ 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 $c'_{k k \oplus 1} = c_{k k \oplus 1} \cdot m$ and $s'_k = \tau(s_k, -m)$. (3) $c_{k \ominus 1 k} = m \cdot c''_{k \ominus 1 k}$ and $\tau(s_k, +m)$ is defined. Then $c'_{k \ominus 1 k} = c''_{k \ominus 1 k}$ and $s'_k = \tau(s_k, +m)$. Except for the changes made above, all other elements of $CF'_{[i..j]}$ remain the same as $CF_{[i..j]}$. The reflexive, transitive closure of $\mapsto_{[i..j]}$ is denoted as $\mapsto^*_{[i..j]}$. $CF'_{[i..j]}$ is *reachable* from $CF_{[i..j]}$ iff $CF_{[i..j]} \mapsto^*_{[i..j]} CF'_{[i..j]}$. We can also define the reachability from $CF_{[i..j]}$ for s_k, (m, s_k), and sending cycle $\mathcal{C}_k$, respectively.

Given $\mapsto^*_{[i..j]}$ among configurations, we have the following two lemmas. The first lemma justifies our finite extension of $\mathbf{F}$ based on configurations, while the second one allows us to perform finite extension on only those configurations indexed by maximal proper incompatible intervals.

Lemma 4.3 Given $CF_{[i..j]}$ and $CF'_{[i..j]}$. Suppose $CF_{[i..j]} \subseteq S$. Then $CF_{[i..j]} \mapsto^*_{[i..j]} CF'_{[i..j]}$ only if there is a state S' such that $S \mapsto^* S'$ and $CF'_{[i..j]} \subseteq S'$.

Lemma 4.4 Suppose $CF_{[i'..j']} \subseteq CF_{[i..j]}$. Then $CF_{[i'..j']} \mapsto^*_{[i'..j']} CF'_{[i'..j']}$ iff $\exists\ CF'_{[i..j]} : (CF'_{[i'..j']} \subseteq CF'_{[i..j]}) \wedge (CF_{[i..j]} \mapsto^*_{[i..j]} CF'_{[i..j]})$.

It should be noted that the set of configurations reachable from $CF_{[i..j]}$ can be infinite. Thus, care must be taken in order to obtain a finite extension of $CF_{[i..j]}$. Denote $CF_{[i..j]} = (c_{i \ominus 1 i}, s_i, .., c_{j \ominus 1 j}, s_j)$. Let $K_k = |c_{k \ominus 1 k}|, k \in [i..j]$. $CF'_{[i..j]}$ satisfies the *channel constraint* w.r.t $CF_{[i..j]}$ iff $\forall k \in [i..j]$: $|c'_{k \ominus 1 k}| \leq MAX(K_k, 1)$. $CF_{[i..j]} \mapsto_{m[i..j]} CF'_{[i..j]}$ iff $CF_{[i..j]} \mapsto_{[i..j]} CF'_{[i..j]}$ and $CF'_{[i..j]}$ satisfies the channel constraint w.r.t $CF_{[i..j]}$. Denote $\mapsto^*_{m[i..j]}$ as the transitive, reflexive closure of $\mapsto_{m[i..j]}$. $CF'_{[i..j]}$ is *m-reachable* from $CF_{[i..j]}$ iff $CF_{[i..j]} \mapsto^*_{m[i..j]} CF'_{[i..j]}$. We can also define the m-reachability from $CF_{[i..j]}$ for s_k, (m, s_k), and sending cycle $\mathcal{C}_k$, respectively. By induction on

the number of transitions executed from $CF_{[i..j]}$ to $CF'_{[i..j]}$, it can be shown that $CF_{[i..j]} \mapsto^*_{m[i..j]} CF'_{[i..j]}$ only if $CF'_{[i..j]}$ satisfies the channel constraint w.r.t $CF_{[i..j]}$.

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

To answer this question, we adopt a technique called *maximal progress configuration exploration* similar to the ones proposed in [3, 7]. Suppose $CF_{[i..j]} \mapsto^*_{[i..j]} CF^d_{[i..j]}$ via local execution sequence set $\{e_i, e_{i\oplus 1}, .., e_j\}$. For any $s^d_k \in CF^d_{[i..j]}, k \in [i..j]$, we want to construct a configuration $CF^b_{[i..k]} = (c^b_{i\oplus 1i}, s^b_i, \ldots, c^b_{k\oplus 1k}, s^b_k)$ such that $(s^d_k = s^b_k) \wedge (CF_{[i..k]} \mapsto^*_{m[i..k]} CF^b_{[i..k]})$ by rearranging the execution order of the transitions in $\{e_i, e_{i\oplus 1}, .., e_k\}$. 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 $CF'_{[i..k]}$. Denote $\vec{\tau}'_{[i..k]} = (\tau'_i, \tau'_{i\oplus 1}, \ldots, \tau'_k)$. For brevity, we define $\mathbf{max}(\vec{\tau}'_{[i..k]}) = l$ if l is the first executable transition in the order from j to i from $\vec{\tau}'_{[i..k]}$ in $CF'_{[i..k]}$; $\mathbf{max}(\vec{\tau}'_{[i..k]}) = 0$ if no such transition can be found in $CF'_{[i..k]}$.

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

It can be shown that the above algorithm does terminate, and when it terminates, $s^b_k = s^d_k$ in $CF^b_{[i..k]}$. By induction on the number of iterations from line 2 to line 6, it can also be shown that $CF_{[i..k]} \mapsto^*_{m[i..k]} CF'_{[i..k]}$. Hence, at the end of the algorithm, $CF_{[i..k]} \mapsto^*_{m[i..k]} CF^b_{[i..k]}$. As for (m, s^d_k), we follow the similar approach used in subsection 4.1. First, we use the above algorithm to get to $CF^b_{[i..k]}$. If $(m, s^d_k) \in CF^b_{[i..k]}$, then we are done. Otherwise, we continue the construction based on the remaining transitions in $\{e_i, e_{i\oplus 1}, \ldots, e_k\}$ until we get to a configuration $CF''_{[i..k]}$ such that m is at the head of channel $C_{k\oplus 1}$. It can be shown such construction is always possible and terminates. Thus, at the end of the construction, $((m, s^d_k) \in CF''_{[i..k]}) \wedge (CF_{[i..k]} \mapsto^*_{m[i..k]} CF''_{[i..k]})$. To summarize, we have the following result:

Theorem 4.1 Given a configuration $CF_{[i..j]}$. Let $k \in [i..j]$. s^d_k is reachable from $CF_{[i..j]}$ iff it is m-reachable from $CF_{[i..k]}$. (m, s^d_k) is reachable from $CF_{[i..j]}$ iff it is m-reachable from $CF_{[i..k]}$. Therefore, both **P-I** and **P-II** are decidable for any configuration via finite extension on $CF_{[i..j]}$.

We would like to point out that the preceding argument for this theorem is brief and informal. The formal proof, however, is quite involved. Please refer to the full paper [6] for details.

4.4 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 reducing the logical error detection problems to two local reachability problems **P-I** and **P-II**. We found out 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 configurations 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 configuration via finite extension. Hence, we have come along to establish the following decidability result:

Theorem 4.2 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 4.3 Given a cyclic protocol $P \in \mathcal{P}$. P is logically correct if **F** does not contain any logical

errors. P has an unspecified reception but $\mathbf{F}_{u,r} = \emptyset$ only if $\mathbf{F}_{u,b} \neq \emptyset$. P is unbounded but $\mathbf{F}_{u,b} = \emptyset$ only if $\mathbf{F}_{u,r} \neq \emptyset$. P has a nonexecutable transition that is not detectable via $\mathbf{F}$ only if $\mathbf{F}_{u,r} \cup \mathbf{F}_{u,b} \neq \emptyset$.

Combined with the result that deadlock detection is decidable via $\mathbf{F}$ [5], 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 the investigation. Furthermore, the decision procedures 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 configurations 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 $\mathbf{F}_{u,r} = \emptyset$ and $\mathbf{F}_{u,b} \neq \emptyset$, and is only performed on those states in $\mathbf{F}_{u,b}$. 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 $\mathbf{MCRG}_k$ is usually small compared to the size of $\mathbf{F}$ due to the channel constraint. Moreover, after $\mathbf{MCRG}_k$ is generated, we can check logical errors in $\mathbf{MCRG}_k$, 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 [2] 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.

5 Conclusion

In this paper, we showed that detection of logical errors other than deadlock are all decidable for $\mathcal{P}$, with possible finite extension of $\mathbf{F}$. Combined with the results established in [5], we have completely solved the logical error detection problems for $\mathcal{P}$. Our study demonstrates that for class $\mathcal{P}$, fair progress state exploration not only can achieve substantial state reduction but also can maintain very competitive logical error detection capability. Therefore, it is a viable state reduction technique.

During our study, we showed the collective power of maximal and fair progress state exploration in producing new results. This should encourage more research on extending or/and combining existing techniques to the validation of more complex protocols. We are currently working on several issues to extend our work: (1) Internal transitions and nondeterminism. (2) More complex and yet regular topologies. (3) Other models such as the extended finite state machine model.

Acknowledgement

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

References

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

[2] 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.

[3] 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.

[4] 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.

[5] H. Liu and R.E. Miller, "Generalized Fair Reachability Analysis for Cyclic Protocols: Part 1," Technical Report CS-TR-3204, Dept. of Computer Science, Univ. of Maryland at College Park, January 1994, in Proc. 14th IFIP Symposium on Protocol Specification, Testing and Verification, Vancouver, B.C. Canada, June 1994.

[6] H. Liu and R.E. Miller, "Generalized Fair Reachability Analysis for Cyclic Protocols: Part 2," *in preparation.*

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

[8] 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.

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