INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

On Exploring
Even Reachable Global State Space
to Verify Deadlock Freedom of Protocols

Tuong M. Nguyen

A Thesis

Submitted to the Faculty of Graduate and Postdoctoral Studies

of the University of Ottawa in Partial Fulfillment of the Requirements for the Degree of

*Master's in Computer Science**

School of Information Technology and Engineering (S.I.T.E.)

Faculty of Engineering

University of Ottawa

Ottawa, Ontario

Canada

---

* The Master's program in Computer Science is a joint program with Carleton University, administered by the Ottawa-Carleton Institute for Computer Science

Acknowledgments

My deep appreciation and respect to my supervisor, Professor Hasan Ural, who introduced me to the field of protocol engineering, and in particular, the reachability analysis problem. Since my undergraduate days at the University of Ottawa, Professor Ural has been continuously motivating and supporting my studies and research. I also would like to thank Dr. Kadir Özdemir and Dr. Hans van der Schoot for the knowledge they shared with me when we were working in the Telecommunications and Software Engineering Research Group (TSERG) lab.

My graduate studies have been funded in part by the Natural Sciences and Engineering Research Council (NSERC), the Faculty of Graduate and Postdoctoral Studies of the University of Ottawa, and the School of Information Technology and Engineering (S.I.T.E.), Faculty of Engineering, University of Ottawa.

Ottawa, April 12, 2002

Abstract

During the fourth international conference on computer communications and networks (ICCCN), which was held in Las Vegas in September 1995, Peng introduced another relief strategy for reachability analysis, called even reachability analysis. Using the CFSM model, the sequentially reachable global states whose total channel length is even form the even reachable global state space, which is a subset of the sequentially reachable global state space. In even reachability analysis, Peng selects the transition pairs that always force a pair of two process transitions to be executed at a time starting from the initial global state of a protocol. Then, the strategy can preserve the even total channel length property in the generated global states and thus, it explores only part of the even reachable global state space to verify deadlock freedom of the protocol. In the proceedings of ICCCN'95, Peng also attempts to prove that deadlock freedom of any n-process protocol with arbitrary topology is verifiable by even reachability analysis with more than one-half reduction in generated global states. This thesis reviews the Peng strategy and proves that the selected set of transition pairs is not sufficient to reach all deadlock states in an n-process protocol with arbitrary topology. Hence, deadlock freedom of the protocol is not verifiable by the Peng strategy for even reachability analysis.

This thesis then forms an extended set of transition pairs to explore the entire even reachable global state space in a finite sequentially reachable global state space. Using the extended set of transition pairs, deadlock freedom of any n-process protocol with arbitrary topology is proved verifiable with around one-half reduction in generated global states. However, the RELIEF tool, which is developed to support the research on reachability analysis in general, (when augmented to implement even reachability analysis) shows negative reduction in global state transitions when using the extended set of transition pairs, especially when the number of processes in a protocol is high. An analytic explanation is also provided in this thesis for that result.

Finally, the ideas of even reachabilty analysis are visited again and compared with existing relief strategies.

Abbreviations

CFSM	Communicating Finite State Machine
CRA	Conventional Reachability Analysis
ERA	Even Reachability Analysis
FIFO	First In, First Out
FRA	Fair Reachability Analysis
ICCCN	International Conference on Computer Communications and Networks
LAN	Local Area Network
LRA	Leaping Reachability Analysis
POVAS	Partial-order Verification based on Ample Sets
RAM	Random Access Memory
SRA	Simultaneous Reachability Analysis
StD	Standard Deviation
iff	if and only if
w.r.t.	with respect to

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Protocol is a set of rules that governs the orderly exchange of messages among interacting processes in a distributed system. One of the most widely used models in specifying and validating communication protocols is the *communicating finite state machine* (CFSM) model [Bo78, BrZa83]. A *global state* of a protocol specified in the CFSM model then consists of a state for each process and the content for each channel. A simple exploration technique, called *reachability analysis*, has been advocated for systematic generation of the *global state space* of the protocol to detect the *logical errors* in the protocol. A protocol is said to be *logically correct* if it does not contain any logical errors. *Deadlock* is one of the logical errors, which occurs at a global state where all channels are empty and none of the processes in the protocol can progress. A protocol is free from deadlock if it does not contain any deadlock states.

Protocols often have a very large global state space and hence exhaustive generation of the global state space by reachability analysis gives rise to the *state explosion* problem. Many strategies have been proposed to relieve the state explosion problem over the last two decades, which analyze only a subset of the entire global state space of a protocol.

Using the CFSM model, even reachability analysis explores the even reachable global state space, which consists of the reachable global states whose total channel length is even, and which is a subset of the reachable global state space of a protocol. In even reachability analysis, a pair of two process transitions is forced to be executed at a time starting from the initial global state of a protocol to generate only the even reachable global states to verify *deadlock freedom* of the protocol. In the proceedings of ICCCN'95 [Pe95], Peng proposed the *selected transition pairs* to explore part of the even reachable

global state space to verify deadlock freedom of any protocol with arbitrary topology. This thesis reviews that paper and proves that the Peng strategy cannot verify deadlock freedom of a protocol with arbitrary topology using the selected transition pairs. The set of transition pairs is then extended to cover the entire even reachable global state space. Empirical study using the RELIEF tool (http://www.site.uottawa.ca/~ural/relief/) is carried out to verify the latter strategy of even reachability analysis in terms of verifying deadlock freedom. The performance results of the strategy are also provided and analytically explained.

The sections in this chapter are arranged as follows: Section 1.1 gives an overview of the work of exploring even reachable global state to verify deadlock freedom of protocols and Section 1.2 lists the main contribution of the thesis.

1.1 Exploring Even Reachable Global State Space to Verify Deadlock Freedom of Protocols

1.1.1 The CFSM Model and Reachability Analysis

One of the most widely used models in specifying and validating communication protocols is the *communicating finite state machine* (CFSM) model [Bo78, BrZa83]. In this model, a communication protocol is specified as network of n, $n \geq 2$, processes that exchange messages over error-free *simplex channels*. Each process is represented by a CFSM and each simplex channel is represented by a FIFO queue. A *global state* of such a protocol then consists of a state for each CFSM and the content for each queue.

The CFSM model lends itself to *reachability analysis* that has been advocated as a *global state space* exploration technique for verification of *logical correctness* of protocols, such as absence of *non-executable transitions*, *deadlocks*, *blocking states*, *buffer overflows*, and *unspecified receptions*. During reachability analysis, all possible reachable global states starting from the initial global state of the protocol are generated and checked against the *logical errors*. Reachability analysis is called *sequential reachability analysis* or *conventional reachability analysis*.

1.1.2 The State Explosion Problem and Relief Strategies

There are two main issues in the use of reachability analysis for the verification of logical correctness of a protocol specified in the CFSM model: *undecidability* and *state explosion*. In general, for an n-process protocol (i.e., a protocol consisting of n, $n \geq 2$, processes) with arbitrary *topology* and arbitrary process structures, the problem of verifying the absence of logical errors is undecidable [BrZa83]. One subclass of protocols for which the verification problem is decidable consists of many practical protocols where all of the channels are bounded [BrZa83].

However, even with *bounded channels*, protocols often have a very large global state space and hence exhaustive generation of the global state space by reachability analysis gives rise to the state explosion problem. Over the last two decades, many *relief strategies* have been proposed to relieve the state explosion problem [LiChLi87, Yu88].

1.1.3 Even Reachable Global State Space and Even Reachability Analysis

Even reachability analysis is a relief strategy that, starting from the initial global state of a protocol with arbitrary topology, forces a pair of two process transitions to be executed at a time. Then, every global states generated by this strategy has an even number of total messages in the queues.

The *even reachable global state space* of a protocol is a subset of the sequentially reachable global state space in which, every global state has an even number of total messages in the queues. Since all channels are empty in a *deadlock state*, it is obvious that every deadlock state in the protocol belongs to the even reachable global state space. However, if an even reachability analysis covers only part of the even reachable global state space, it must be proved that each deadlock state is reachable by that strategy. Otherwise, the strategy cannot verify deadlock freedom of the protocol.

During the fourth international conference on computer communications and networks (ICCCN'95), which was held in Las Vegas in September 1995, Peng proposed the *selected transition pairs* to explore only part of the even reachable global state space

to verify deadlock freedom of a protocol. This thesis reviews the Peng strategy and proves that the selected set of transition pairs is not sufficient to reach all deadlock states in a protocol with arbitrary topology. Hence, deadlock freedom of the protocol is not verifiable by the Peng strategy for even reachability analysis.

An extended set of transition pairs is experimented to explore the entire even reachable global state space in a finite reachable global state space. Using the extended set of transition pairs, deadlock freedom of any protocol having more than two processes with arbitrary topology is proved verifiable with around one-half reduction in generated global states. However, the RELIEF tool, which is developed to support the research on reachability analysis in general, shows negative reduction in global state transitions when using the extended set, especially when the number of processes in a protocol is high. An analytic explanation is also provided in this thesis for that result.

1.2 Main Contributions of the Thesis

This thesis mainly discusses two relief strategies of even reachability analysis:

- One is called *ERAp*, which was proposed by Peng [Pe95] during the fourth international conference on computer communications and networks (ICCCN'95) held in Las Vegas in September 1995.
- One is called *ERAf*, which is a new relief strategy proposed in this thesis.

The main contributions of this thesis are listed in the following two subsections:

1.2.1 The ERAp Strategy

- Rewriting the ERAp strategy to the best of its intents using the terminology that we adopted for the CFSM model.
- Criticizing the inaccuracy in [Pe95].
- Implementing the ERAp strategy in the RELIEF tool.

- Proving the failure of the ERAp strategy theoretically and empirically.
  - Pointing out the incorrectness in the mathematical proof of the main theorem in [Pe95].
  - Presenting a counter-example to show that ERAp cannot verify deadlock freedom of a protocol with arbitrary topology.
  - Showing that ERAp is not a generalization of fair reachability analysis.

1.2.2 The ERAf Strategy

- Formulating the ERAf strategy using the above set of notations.
- Contrasting the ERAf strategy and the ERAp strategy.
- Contrasting the ERAf strategy with fair reachability analysis.
- Proving mathematically that ERAf can verify deadlock freedom of a protocol with arbitrary topology.
- Implementing the ERAf strategy in the RELIEF tool.
- Running an empirical study to verify that deadlock freedom of a protocol with arbitrary topology is decidable by ERAf.
- Reporting the reductions by ERAf in the number of global states and global transitions w.r.t. conventional reachability analysis.
- Explaining analytically the above reductions.
- Experimenting a reduced version of ERAf.

The remaining chapters of this thesis are arranged as follows: Chapter 2 prepares the rules of using CFSM model to specify a communication protocol and

introduces the logical errors; Chapter 3 recalls the state explosion problem and gives an overview of some relief strategies that are related to the work for this thesis; Chapter 4 presents the RELIEF tool; Chapter 5 reviews the ERAp strategy in [Pe95]; Chapter 6 proposes the ERAf strategy; Chapter 7 is for the concluding remarks of the thesis and Chapter 8 lists the references.

Chapter 2 The CFSM Model

Protocol is a set of rules that governs the orderly exchange of messages among interacting processes in a distributed system. Although this definition implies that protocols are related to all areas where interaction between processes is inherent, this thesis merely means communication protocols.

One of the most widely used models in specifying and validating communication protocols is the *communicating finite state machine* (CFSM) model [Bo78, BrZa83]. In this model, a communication protocol is specified by a set of n, $n \geq 2$, processes exchanging messages over error-free, bounded, *simplex channels*. Each process is represented by a CFSM and each simplex channel is represented by a FIFO queue. Each CFSM then communicates asynchronously with the other CFSMs by sending and receiving messages over the FIFO queues. The CFSM model can be used to represent a large number of protocols with arbitrary *topology*, including the *multi-cyclic* protocols that are composed of multiple disjoint *unidirectional rings* [ScUr95c].

A state of a protocol consists of the state of each process and the content of each channel in the protocol. A state of a protocol is so called a *global state*. Some types of *logical errors* that may occur at a global state are specified as follows:

- A *deadlock* occurs at a global state when all channels are empty and none of the processes in the protocol is ready to send any messages.
- A *blocking state* occurs at a global state when none of the processes in the protocol is ready to send or receive any messages.
- A *buffer overflow* channel occurs at a global state when a process attempts to send a message over a channel that is already full.

- An *unspecified reception* occurs at a global state when the head of a message queue in some non-empty channel cannot be received by the corresponding process.

This chapter prepares the rules of using CFSM model to specify a communication protocol and introduces the logical errors. The technique to validate the protocol by verifying the absence of logical errors will be presented in the next chapter. The sections in this chapter are arranged as follows: Section 2.1 defines the CFSM preliminaries that are used to model the example protocols given in Section 2.2.

2.1 The CFSM Preliminaries

Definition 2.1 Let $I=\{1, 2, \ldots, n\}$ be a finite index set. A *protocol* Π is a pair (P, C), where

- $P=\{P_i | i \in I\}$ is a set of n, $n \geq 2$, *processes*.
- $C=\{C_{ij} | i, j \in I \wedge i \neq j\}$ is a non-empty set of error-free, bounded, *simplex channels* between different processes.

Each *process* $P_i \in P$ is a quadruple $(S_i, s_i^0, M_i, \delta_i)$, where

- S_i is a finite, non-empty set of *process states*.
- $s_i^0 \in S_i$ is the *initial state*.
- $M_i=\{x | x \in M_{ij} \cup M_{ji}, \forall j \in I \wedge j \neq i\}$ is a finite, non-empty set of *messages*. Each M_{ij} represents the messages that process P_i can send to process P_j over channel C_{ij}.
- δ_i: $S_i \times M_i \rightarrow S_i$ is a partially defined *process transition function*.

Each error-free, bounded, *simplex channel* C_{ij} is represented by a FIFO queue linking process P_i to process P_j.

- P_i is called the *sender process* and P_j is called the *receiver process* w.r.t. channel C_{ij}.
- C_{ij} is called an *input channel* of process P_j and an *output channel* of process P_i.
- The content c_{ij} of channel C_{ij} is a finite sequence of messages in transit from process P_i to process P_j. The *length of channel* C_{ij} is defined by the number of messages in the content of C_{ij} and is denoted by $|c_{ij}|$. If channel C_{ij} is empty, its content is conventionally represented by ε, i.e., $\forall C_{ij} \in C\ (c_{ij}=\varepsilon \Leftrightarrow |c_{ij}|=0)$.
- The maximum number of the messages in transit that channel C_{ij} can accommodate is called the capacity of channel C_{ij} or *channel bound* B_{ij} of channel C_{ij}, i.e., $\forall C_{ij} \in C\ (|c_{ij}| \leq B_{ij})$. □

Based on these primary definitions, further definitions that are provided in the rest of this section continue to add more meanings to the CFSM model, making it an unambiguous source for the protocol validation techniques. Notice that Definition 2.1 defines a bounded protocol, i.e., the global state space of the protocol is finite.

Definition 2.2 In process P_i of protocol Π, a triple $(s_i, x, \delta_i(s_i, x))$ with $s_i \in S_i$ and $x \in M_i$ is said to be a *process transition* t iff $\delta_i(s_i, x)$ is defined. Specifically,

- If $x \in M_{ij}$, then the transition is called a *sending transition* σ_{ij} and represents the transmission of message x from process P_i to process P_j.
- If $x \in M_{ji}$, then the transition is called a *receiving transition* ρ_{ji} and represents the reception of message x from process P_j by process P_i.

For readability purposes, a minus sign, −, is used to identify a message transmission and plus sign, +, is used to identify a message reception. □

Definition 2.3 A channel is said to be *active* iff there exists a sending transition defined in the sender process w.r.t. the channel. The set of active channels in a protocol is denoted by $active(C)$, i.e.,

$$\forall i, j \in I \wedge i \neq j \; (C_{ij} \in active(C) \Leftrightarrow \exists \delta_i(s_i, x) \wedge x \in M_{ij}).$$ □

Definition 2.4 The *topology*, denoted by T, of protocol Π is a pair $(P, active(C))$ represented by the *communication topology graph* that is in fact a directed graph, also denoted by T, of n nodes, where each node represents a process of protocol Π and each arc represents an active channel of protocol Π. □

Definition 2.5 Protocol Π is *multi-cyclic* iff its communication topology graph T is composed of a finite, non-empty set of *unidirectional rings* $R=\{r_u | u \in \{1, ..., m\} \wedge m \geq 1\}$, where r_u is an ordered set of channels oriented in the same direction forming a closed loop, and all of the following conditions are satisfied:

1. T is strongly connected,
2. $\forall u, v \in \{1, ..., m\} \; (u \neq v \Rightarrow r_u \cap r_v = \varnothing)$ and
3. $\forall i, j, k, l \in I, \forall u \in \{1, ..., m\} \; (C_{ij}, C_{kl} \in r_u \Rightarrow B_{ij} = B_{kl})$. □

Conditions 1 and 2 in Definition 2.5 together state that each simplex channel in a multi-cyclic protocol pertains to exactly one ring in the protocol, a requirement that implicitly confines the multiple ring topology such that, no two rings in the protocol share a common simplex channel. Condition 3 states that all simplex channels pertaining to a given ring in a multi-cyclic protocol must have equal capacity. Notice that Definition 2.5 accounts for the cyclic protocols studied in [LiMi93, LiMi94a, LiMi94b, LiMi96] and that it has, in fact, a rather broad applicability in practice [ScUr95a, ScUr95b, ScUr96a]. In addition to a ring topology, the link structure of a multi-cyclic protocol allows the modeling of a number of common topologies for local area networks (LAN), such as daisy-chain, a star, a tree as shown in Figure 2.1, and also many more combinations of these elementary topologies.

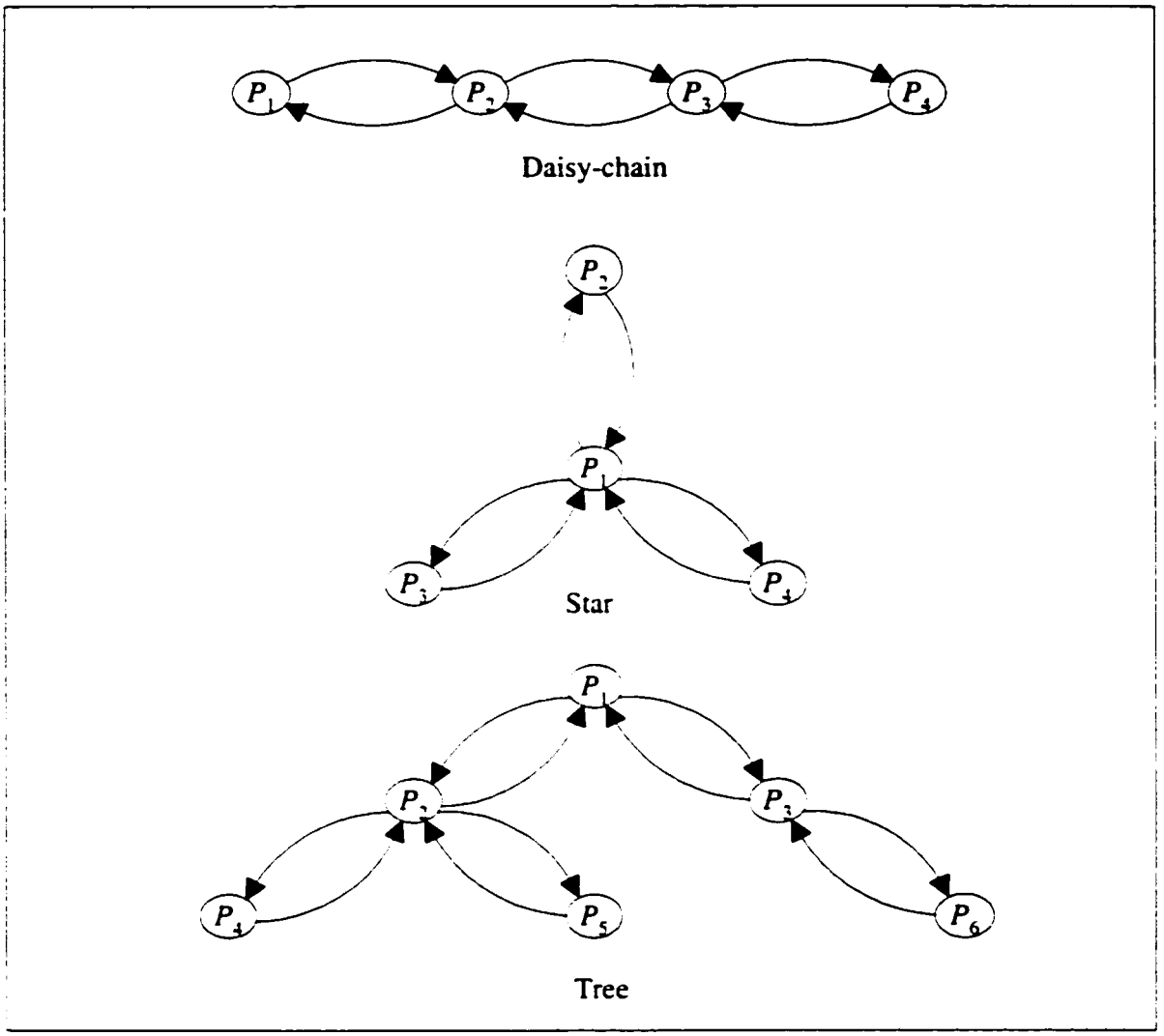

Figure 2.1 Some common topologies for LAN

Definition 2.6 A *global state* G_k of protocol Π is a pair $(\langle s_i^k \rangle_{i \in I}, \langle c_{ij}^k \rangle_{i,j \in I \wedge i \neq j})$ such that s_i^k is the state of P_i and $c_{ij}^k \in M_{ij}^*$ is the content of channel C_{ij} at G_k where M_{ij}^* denotes the set of all finite sequences of messages from M_{ij} including the empty sequence ε. G_0 denotes the *initial global state* of protocol Π where each process is at its initial state and all channels are empty, i.e., $G_0=(\langle s_i^0 \rangle_{i \in I}, \langle c_{ij}^0 \rangle_{i,j \in I \wedge i \neq j})$ and $c_{ij}^0=\varepsilon$. □

spec(s_i), *spec*(G_k), *spec*(P_i), and *spec*(P) denote all transitions at a process state, at a global state, in a process and in a set of processes, respectively, i.e.,

1. $spec(s_i)=\{t \mid t=(s_i, x, \delta_i(s_i, x)) \wedge \delta_i(s_i, x)$ is defined in $P_i\}$,
2. $spec(G_k)=\cup_{i\in I} spec(s_i^k)$,
3. $spec(P_i)=\{t \mid s_i\in S_i \wedge t\in spec(s_i)\}$ and
4. $spec(P)=\{t \mid P_i\in P \wedge t\in spec(P_i)\}$, respectively.

Definition 2.7 A global state G_l is a *sequential immediate successor* of a global state G_k, denoted by $G_k\rightarrow G_l$ or $G_k\text{—}t\rightarrow G_l$, iff $\exists t\in spec(G_k)$ such that one of the following conditions is satisfied:

1. $t=(s_i^k, -x, s_i^l) \wedge s_i^l=\delta_i(s_i^k, -x) \wedge c_{ij}^l=c_{ij}^k x \wedge |c_{ij}^k|<B_{ij} \wedge \forall h\in I\ (h\neq i \Rightarrow s_h^l=s_h^k) \wedge \forall h, m\in I \wedge h\neq m\ (\neg(h=i \wedge m=j) \Rightarrow c_{hm}^l=c_{hm}^k)$.
2. $t=(s_i^k, +x, s_i^l) \wedge s_i^l=\delta_i(s_i^k, +x) \wedge c_{ij}^k=xc_{ij}^l \wedge \forall h\in I\ (h\neq i \Rightarrow s_h^l=s_h^k) \wedge \forall h, m\in I \wedge h\neq m\ (\neg(h=i \wedge m=j) \Rightarrow c_{hm}^l=c_{hm}^k)$. □

Those conditions correspond to a message transmission and a message reception by a process, respectively. Then, the sequential immediate successor defines a binary relation between global states, which is denoted by $\rightarrow$.

Definition 2.8 Let $\rightarrow^*$ be the reflexive and transitive closure of $\rightarrow$.

1. G_m is sequentially reachable from G_k (or a sequential successor of G_k) iff $G_k\rightarrow^* G_m$.
2. G_m is sequentially reachable iff $G_0\rightarrow^* G_m$.

$G_k\text{—}t_1\rightarrow G_{k(1)} \wedge G_{k(1)}\text{—}t_2\rightarrow G_{k(2)} \wedge \ldots \wedge G_{k(j-1)}\text{—}t_j\rightarrow G_m$ is also denoted by $G_k\text{—}t_1t_2\ldots t_j\rightarrow^* G_m$. □

Definition 2.9 For any $i, j\in I$ and $i\neq j$, a process transition $t\in spec(P_i)$ is executable if there exists a sequentially reachable global state G_k such that one of the following two conditions is satisfied:

1. $t=(s_i^k, -x, \delta_i(s_i^k, -x)) \wedge x \in M_{ij} \wedge |c_{ij}^k| < B_{ij}$.

2. $t=(s_i^k, +x, \delta_i(s_i^k, +x)) \wedge x \in M_{ji} \wedge c_{ji}^k = xX \wedge X \in M_{ji}^*$.

The set of all executable transitions at global state G_k is denoted by $exec(G_k)$. □

Definition 2.10 In protocol Π, *deadlocks*, *blocking states*, *buffer overflows*, *unspecified receptions*, and *non-executable transitions* are *logical errors* and defined as follows:

1. A process transition t is a *non-executable transition* in protocol Π if $t \in spec(P)$ and t is not executable.

Let $G_0 \rightarrow^* G_k$. At global state G_k,

2. A *deadlock* occurs if $exec(G_k)=\varnothing \wedge \forall i, j \in I \wedge i \neq j\ c_{ij}=\varepsilon$.

3. A *blocking state* occurs if $exec(G_k)=\varnothing$.

4. A *buffer overflow* occurs in channel C_{ij} if $(s_i^k, -x, \delta_i(s_i^k, -x)) \in spec(G_k) \wedge x \in M_{ij} \wedge |c_{ij}^k| = B_{ij})$.

5. An *unspecified reception* of $x \in M_{ij}$ occurs if $c_{ij}^k = xX \wedge X \in M_{ij}^* \wedge \delta_j(s_j^*, -x)$ is not defined. □

A non-executable transition is a protocol-based logical error. A deadlock, a blocking state, a buffer overflow, and an unspecified reception are global state-based logical errors, i.e., each of them if exists can be identified immediately once a global state and the CFSMs are given. A deadlock is obviously a blocking state, but the reverse is not always true. A buffer overflow or an unspecified reception can be a blocking state but not always.

Definition 2.11 For any $i, j \in I$ and $i \neq j$, a transition $t \in spec(P_i)$ is called a *potentially executable transition* at global state G_k iff one of the following conditions is satisfied:

1. $t=(s_i^k, -x, \delta_i(s_i^k, -x)) \wedge x \in M_{ij} \wedge |c_{ij}^k| = B_{ij}$.

2. $t=(s_i^k, +x, \delta_i(s_i^k, +x)) \wedge x \in M_{ji} \wedge c_{ij}^k=\varepsilon$.

Condition 1 specifies a *potentially executable sending transition* and Condition 2 specifies a *potentially executable receiving transition*. The set of all potentially executable transitions at global state G_k is denoted by *potent*(G_k). □

Conceptually, a potentially executable sending transition at global state G_k is a message transmission that is not executable just because the channel is already full. However, it will become executable at a successor of G_k if the receiver process at the other end of the channel receives a message, i.e., removes the head message from the corresponding FIFO queue.

Similarly, a potentially executable receiving transition at global state G_k is a message reception that is not executable just because the channel is empty. However, it will become executable at a successor of G_k if the sender process at the other end of the channel sends exactly the expected message to the corresponding FIFO queue. A potentially executable receiving transition is sometimes called an *enabled transition* [LiMi94a].

2.2 The CFSM Examples

Example 2-1 In this example, protocol Π1 has two processes exchanging messages over two error-free, bounded, simplex channels. Each channel can accommodate at most two messages.

Figure 2.2 shows the topology of protocol Π1 and the CFSMs of its two processes. Protocol Π1 is the simplest case of multi-cyclic protocols, which consists of only one unidirectional ring of only two channels.

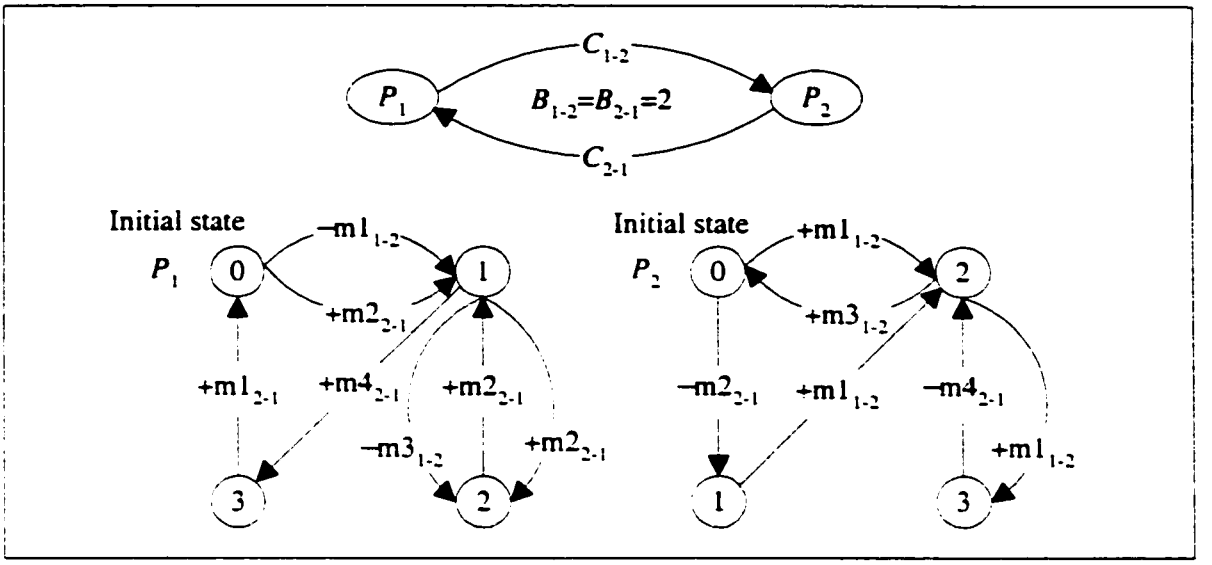

Figure 2.2 The CFSM model of two-process protocol Π1

$P=\{P_1, P_2\}$ and $active(C)=\{C_{1-2}, C_{2-1}\}=C$

$S_1=\{0, 1, 2, 3\}$	$S_2=\{0, 1, 2, 3\}$
$s_1^0=0$	$s_2^0=0$
$M_{1-2}=\{m1, m3\}$	$M_{2-1}=\{m2, m4\}$
$M_1=\{m1, m2, m3, m4\}$	$M_2=\{m1, m2, m3, m4\}$
$\delta_1(0, -m1)=1$	$\delta_2(0, -m2)=1$
$\delta_1(0, +m2)=1$	$\delta_2(0, +m1)=2$
$\delta_1(1, -m3)=2$	$\delta_2(1, +m1)=2$
$\delta_1(1, +m2)=2$	$\delta_2(2, +m3)=0$
$\delta_1(1, +m4)=3$	$\delta_2(2, +m1)=3$
$\delta_1(2, +m2)=1$	$\delta_2(3, -m4)=2$
$\delta_1(3, +m1)=0$	

□

Example 2-2 Protocol Π2 in this example has four processes exchanging messages over four error-free, bounded, simplex channels as shown in Figure 2.3.

$P=\{P_1, P_2, P_3, P_4\}$ and $active(C)=\{C_{1\text{-}2}, C_{2\text{-}3}, C_{3\text{-}4}, C_{4\text{-}3}\}\subset C$

$B_{1\text{-}2}=B_{2\text{-}3}=B_{3\text{-}4}=B_{4\text{-}3}=5$

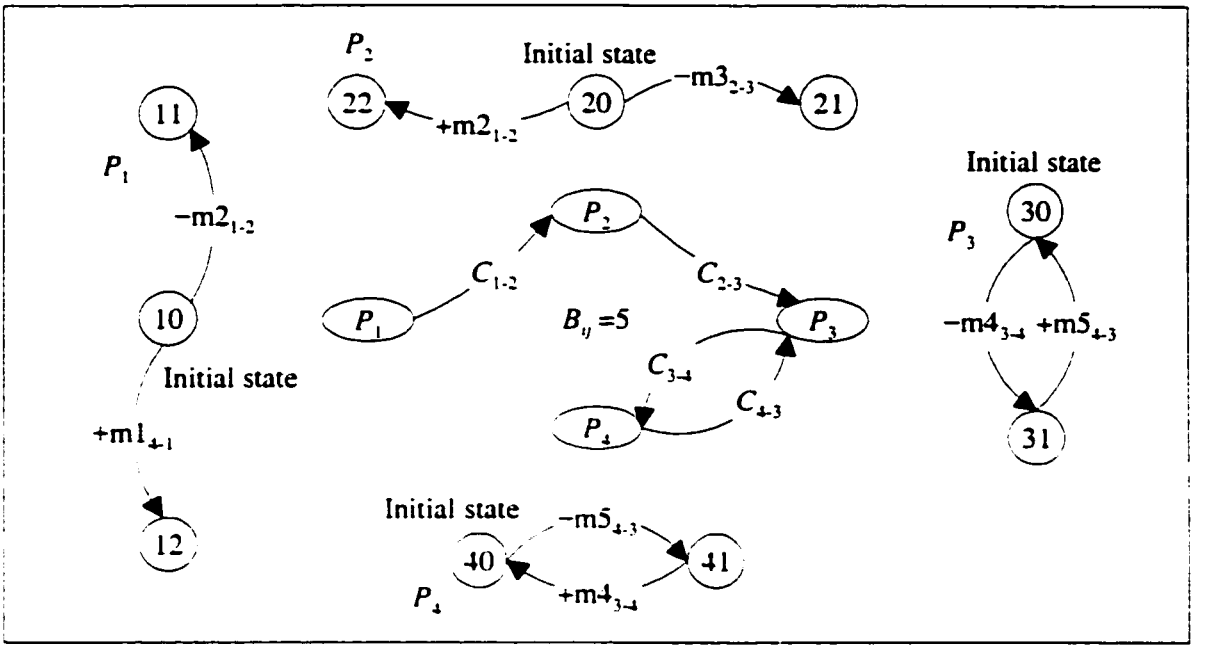

Figure 2.3 The CFSM model of four-process protocol Π2

A global state G_k of protocol Π2 is represented by a pair $G_k=(<s_1^k, s_2^k, s_3^k, s_4^k>,<c_{1\text{-}2}^k, c_{2\text{-}3}^k, c_{3\text{-}4}^k, c_{4\text{-}3}^k>)$. The initial global state of Π1 is $G_0=(<10, 20, 30, 40>,<\varepsilon, \varepsilon, \varepsilon, \varepsilon>)$. Observe that not all channels in protocol Π2 are active. For instance, channel $C_{4\text{-}1}$ is not active, by Definition 2.3, since transition $\delta_4(s_4^k, -m1)$ is not defined in process P_4 for any $s_4^k \in S_4$, $S_4=\{40, 41\}$. The receiving transition $(10, +m1_{4\text{-}1}, 12)$ in process P_1 must be a non-executable transition in this case. □

Example 2-3 In this example, protocol Π3 has four processes and four active channels, which compose a unidirectional ring. Protocol Π3 is still a simple case of multi-cyclic protocols, which consists of only one unidirectional ring. Figure 2.4 shows the topology of protocol Π3 and the CFSMs of its four processes.

$P=\{P_1, P_2, P_3, P_4\}$ and $active(C)=\{C_{1\text{-}2}, C_{2\text{-}3}, C_{3\text{-}4}, C_{4\text{-}1}\}$

$B_{1\text{-}2}=B_{2\text{-}3}=B_{3\text{-}4}=B_{4\text{-}1}=5$

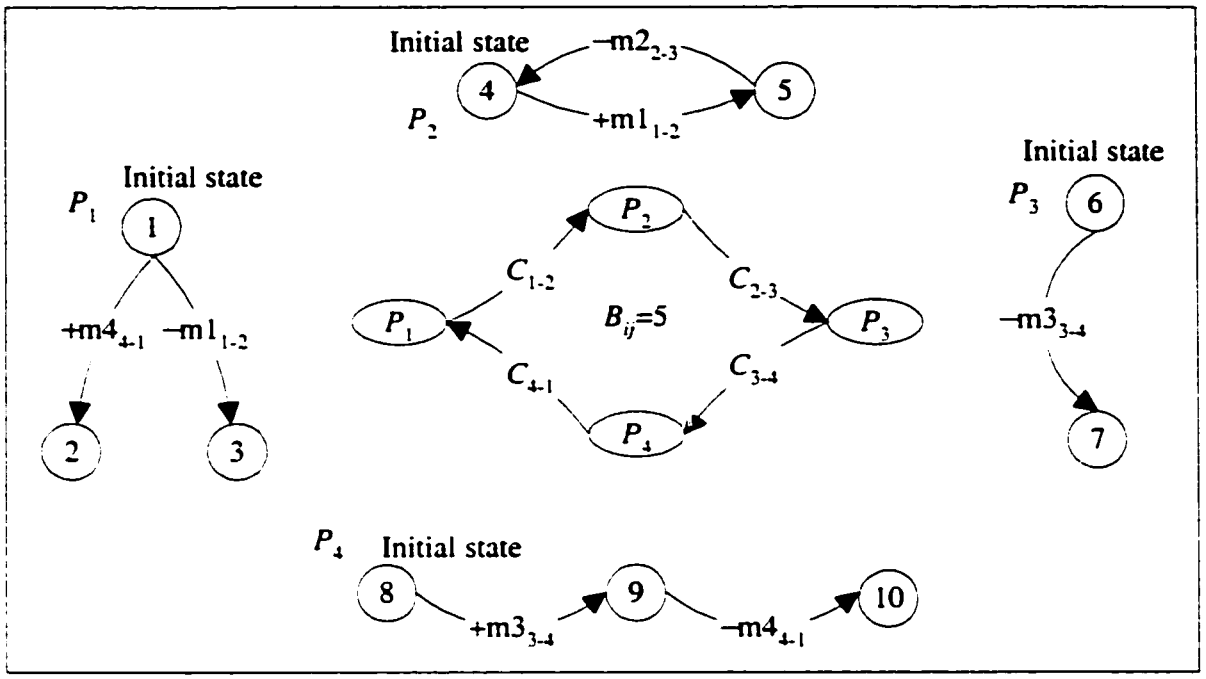

Figure 2.4 The CFSM model of cyclic protocol Π3

Of course, a multi-cyclic protocol with only one unidirectional ring can be simply called a *cyclic* protocol. Notice that, each process in a cyclic protocol has exactly one input channel and only one output channel. □

Example 2-4 Multi-cyclic protocol Π4 in this example is composed of two unidirectional rings, r_1 and r_2, as shown in Figure 2.5.

$P=\{P_1, P_2, P_3, P_4\}$ and $active(C)=\{C_{1\text{-}2}, C_{2\text{-}3}, C_{3\text{-}1}, C_{3\text{-}4}, C_{4\text{-}3}\} \subset C$

$r_1=\{C_{1\text{-}2}, C_{2\text{-}3}, C_{3\text{-}1}\}$, $r_2=\{C_{3\text{-}4}, C_{4\text{-}3}\}$

$B_{1\text{-}2}=B_{2\text{-}3}=B_{3\text{-}1}=2$ and $B_{3\text{-}4}=B_{4\text{-}3}=5$

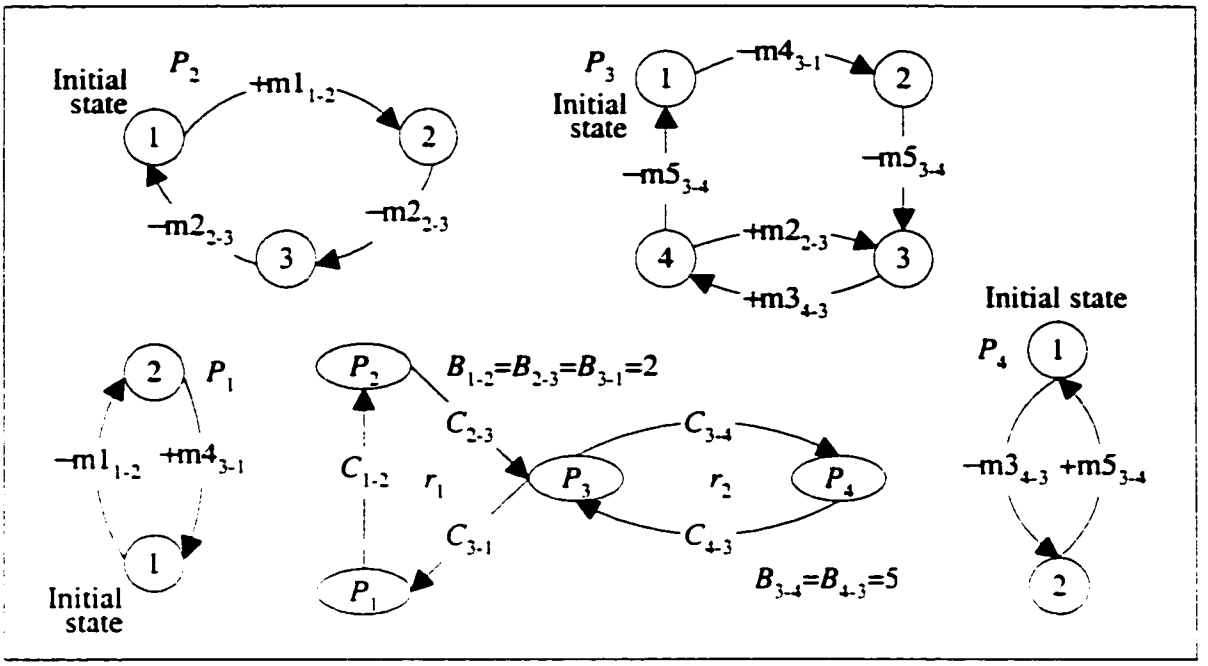

Figure 2.5 The CFSM model of multi-cyclic protocol Π4

Recall that, each of the four active simplex channels in protocol Π4 must pertain to exactly one unidirectional ring, and all simplex channels pertaining to a given ring in a multi-cyclic protocol must have equal capacity. In a multi-cyclic protocol, a process can participate in more than one unidirectional ring, then the number of its input channels and the number of its output channels must be equal. □

Many methodologies can be used to specify a communication protocol. Beside the CFSM model, some of the most popular are the Petri Net model [Da80], the Language of Temporal Ordering Specification (LOTOS) [LoFaHa92], the Specification and Description Language (SDL) [BeHo89], and the Extended State Transition Language (Estelle) [CoPeAy85, Li85]. In this chapter, the CFSM model has been shown simple but powerful enough to specify a large number of protocols with arbitrary topology. The constructions of CFSMs, active channels, communication topology graph, and global states for a protocol are all straightforward. Except non-executable transitions, other

logical errors are global state-based. Then, validating a communication protocol is nothing but generating and analyzing every global state to verify the absence of logical errors. Each process transition is marked when it is executed, and the non-executable transitions can be identified at the end of the exploration. Those are the main ideas of the *reachability analysis* technique to be presented in Chapter 3.

Chapter 3 The Reachability Analysis Technique

A simple exploration technique, called *reachability analysis*, has been advocated for systematic generation of the *global state space* of a protocol specified in the CFSM model to detect non-executable transitions and erroneous global states such as deadlocks, blocking states, buffer overflows and unspecified receptions. The existence of an erroneous global state or a non-executable transition is called a logical error in a protocol. A protocol is said to be *logically correct* if it does not contain any logical errors.

Notice that the logical correctness of a protocol does not imply the functional correctness of the protocol. A protocol may be functionally incorrect but may not include any logical error. There is an analogy between the syntactic correctness of a program and the logical correctness of a protocol. The logical errors are sometimes called *syntactic errors* of a protocol [Öz95].

There are two main issues in the use of reachability analysis for the verification of logical correctness of a protocol specified in the CFSM model: *undecidability* and *state explosion*. In general, for an n-process protocol (i.e., a protocol consisting of n, $n \geq 2$, processes) with arbitrary topology and arbitrary process structures, the problem of verifying the absence of logical errors is undecidable [BrZa83]. Decidability of the logical correctness problem for subclasses of protocols is reported in [YuGo82, BrZa83, Pa87, Fi88, LiMi94a, LiMi94b]. One subclass of protocols for which the verification problem is decidable consists of many practical protocols where all of the channels are bounded [BrZa83]. By including the definition of channel bound in the CFSM preliminaries (Definition 2.1), the issue of undecidability is bypassed in the context of this thesis.

However, even with bounded channels, protocols often have a very large global state space and hence exhaustive generation of the global state space by reachability analysis gives rise to the state explosion problem. Many strategies have been proposed to relieve the state explosion problem, which can be roughly categorized into two main groups. One group of relief strategies makes some restricting assumptions, such as

- The simplex channels have length zero [Bo78].
- The number of processes is two [RuWe82, GoYu84, GoHa85].
- The processes do not contain cycles other than the ones passing through their initial states [ItIc83].
- The processes are in certain structures [VuCo82, ChMi83].
- A process at a state can either transmit or receive a message [YuGo82].
- The topology of the protocol is a unidirectional ring [LiMi93, LiMi94a, LiMi94b, LiMi96].
- The topology of the protocol is multi-cyclic [ScUr95a, ScUr95b, ScUr95c, LiMiScUr96].

The other group of relief strategies analyzes only a subset of the global state space of the protocol, such as

- The subset of global state space reachable by simultaneous execution of transitions [ItIc83, ÖzUr94, ÖzUr95, ScUr95d, ScUr96b, ScUr96c, ScUr97, ScUr98a].
- The subset of global state space associated with some channel length property [LiMi93, LiMi94a, LiMi94b, LiMi96, ScUr95a, ScUr95b, ScUr95c, Pe95].
- Others [Ho88, MaSa87, We86].

This chapter recalls the global state explosion problem, gives an overview of some relief strategies that are related to the work for this thesis. The sections in this chapter are then arranged as follows: Section 3.1 describes the conventional reachability analysis and the global state explosion problem, then Section 3.2 reviews the relief strategies.

3.1 Conventional Reachability Analysis

Definition 3.1 A protocol Π is logically correct if it does not contain any logical error. □

In *sequential reachability analysis* [Bo78], which is interchangeably called *conventional reachability analysis* in this thesis, logical errors in a given protocol are determined by generating all sequentially reachable global states from the initial global state with either a transmission or a reception of a message by a process at each step, following Definition 2.7 and Definition 2.8. If a global state includes a logical error, the error type and at least one transition sequence from the initial global state to the erroneous global state are reported. During this analysis, a global state may be generated several times and thus to guarantee the termination of the analysis, the repeated occurrences of each global state must be recognized not to be analyzed again. An algorithm for conventional reachability analysis is presented in Figure 3.1.

Γ is the set of global states that have been checked against logical errors and W is the set of global states to be checked. The algorithm then performs an exploration in the conventional global state space. During the exploration, all transitions from the current global state are explored, and therefore all the sequential immediate successors of the current global states are generated before they are analyzed. As pointed out in [Ho92], if W is a stack then the algorithm performs a depth-first search, and if W is a queue then the algorithm performs a breadth-first search.

If the breadth-first search is used, the global states are analyzed in the order they are generated (or reached). Breadth-first search algorithm guarantees that all global states are reached by one of the shortest paths leading them from the initial global state.

Therefore, erroneous global states are discovered by the shortest execution sequences in this search.

```
Γ=∅        /* Γ is the set of global states already analyzed */
W={G0}     /* W is the set of global states to be analyzed */
while W≠∅ do
begin
    remove an element Gk from W
    add Gk to Γ
    if Gk is a blocking state then
        report blocking state
        if Gk is a deadlock state then report deadlock
    for each transmission (s_t^k, −x, s_t^{k-1}) ∈ spec(Gk) where x∈M_ij and i≠j do
        if |c_ij^k|=B_ij then report buffer overflow
    for each c_ij^k (i≠j) do
        if c_ij^k=xX, X∈M_ij* and δj(s_j^k, +x) is undefined then
            report unspecified reception
    for each Gm such that Gk→Gm do
        if (Gm is not in Γ or W) add Gm to W
end
report each non-executed transition as a non-executable transition
```

Figure 3.1 An algorithm for conventional reachability analysis

If the depth-first search is used, erroneous global states are not likely to be reached by the shortest paths. However, unlike the breadth-first search, the depth-first search algorithm does not need to store information about successor relation between global states to report the paths from the initial global state to the erroneous ones because the global states on the path are currently stored in the stack. In addition, the depth-first

search algorithm can minimize the memory requirements at the expense of running time by storing only global states in a single execution path from the initial global state to the current global state. Since such an algorithm cannot determine, whether a newly generated global state was encountered before in a previous explored path, the algorithm may redundantly analyzes the same global state several times. Variations of the depth-first search algorithms are studied in [Ho92].

Definition 3.2 The *sequentially reachable global state space* denoted by Γ, which is a set of all global states reachable by conventional reachability analysis technique, can be represented by a directed graph, called *conventional reachability graph* and also denoted by Γ in which, the nodes represent global states and the arcs stand for the immediate successor relation between the states, which will be also called *global transitions* in this thesis. □

Two-process protocol Π1 specified in Figure 2.2 of Example 2-1 includes various types of logical errors. Figure 3.2 shows the conventional reachability graph of this protocol, which has twenty-eight global states (i.e., $|\Gamma|=28$) and thirty-eight global transitions.

A global state G_k of protocol Π1 is represented by a pair $G_k=(<s_1^k, s_2^k>,<c_{1\text{-}2}^k, c_{2\text{-}1}^k>)$. The initial global state of Π1 is $G_0=(<0, 0>,<\varepsilon, \varepsilon>)$. Notice that in applications only, a period, ., is used as the separator for the messages in a queue, and a dash, -, is used as the separator for the process indices when specifying a channel.

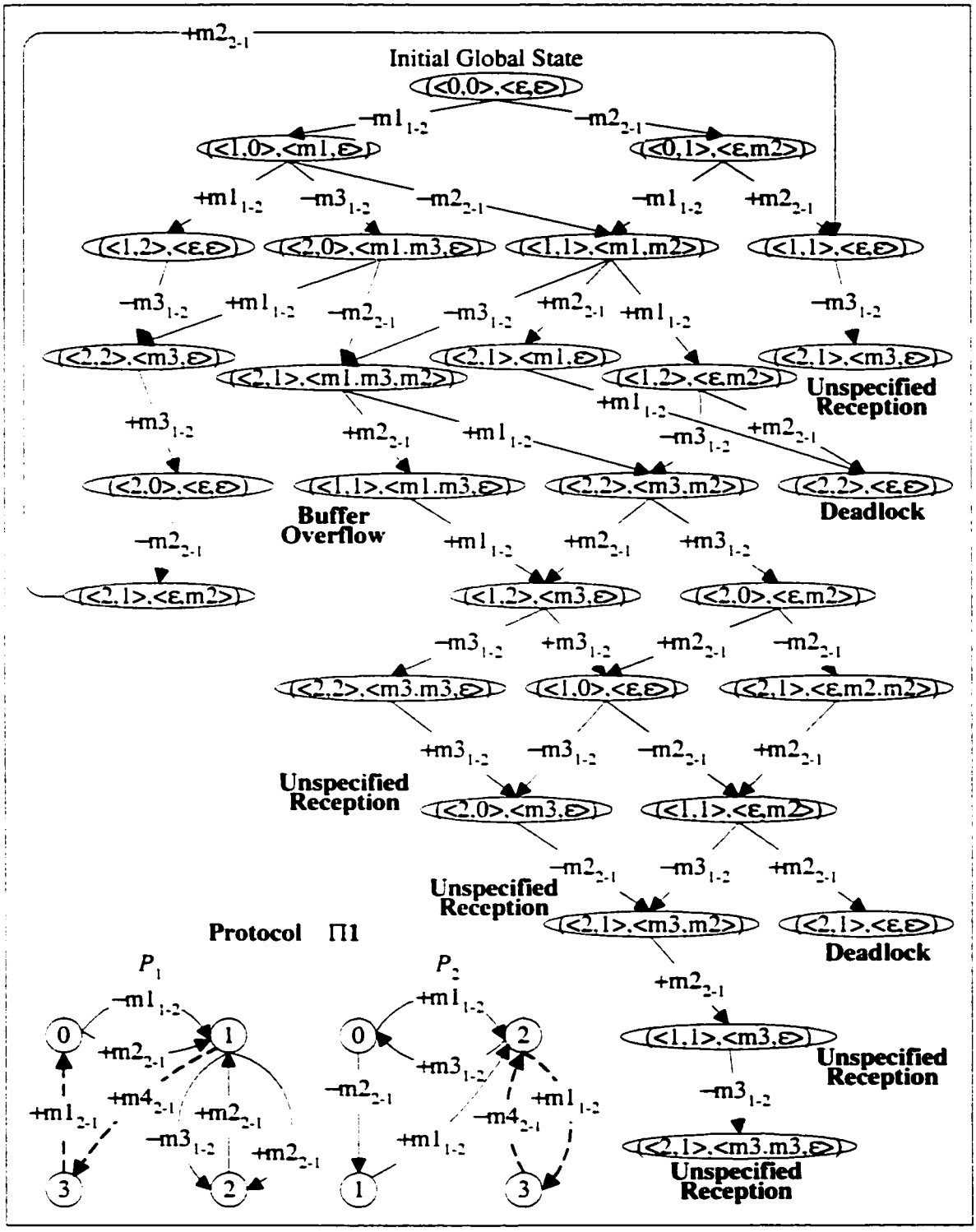

Figure 3.2 The conventional reachability graph of two-process protocol Π1

Protocol Π1 has some logical errors such as

- Two deadlocks occurring at global states (<2, 1>,<ε, ε>) and (<2, 2>,<ε, ε>).
- A blocking state occurring at global state (<2, 1>,<m3, ε>). This is not a deadlock since channel $C_{1,2}$ is not empty at this global state, however an unspecified reception occurs at this global state. Another blocking state is (<2, 1>,<m3.m3, ε>) at which an unspecified reception occurs.
- A buffer overflow occurring in channel $C_{1,2}$ at global state (<1, 1>,<m1.m3, ε>). This is not a blocking state since transition $\delta_2(1, +m1)=2$ defined in process P_2 is ready to be executed.
- An unspecified reception of message m3 in channel $C_{1,2}$ occurring at global state (<2, 0>,<m3, ε>) because transition $\delta_2(0, +m3)$ is not defined in process P_2. This is not a blocking state since transition $\delta_2(0, -m2)=1$ defined in process P_2 is ready to be executed. Other unspecified receptions occur at global states (<2, 1>,<m3, m2>) and (<1, 1>,<m3, ε>).

Each deadlock, buffer overflow, and unspecified reception is marked down on the conventional reachability graph of this protocol. The blocking states can be evidently identified from the graph as well. Since transitions (1, +m4, 3) and (3, +m1, 0) in process P_1, as well as transitions (2, +m1, 3) and (3, −m4, 2) in process P_2 are never executed during the conventional reachability analysis of this protocol, then they are the non-executable transitions of the protocol and shown by dashed lines on the CFSMs of protocol Π1, which is enclosed in Figure 3.2.

Reachability analysis is well suited for verifying protocols against their logical errors, but it often suffers from *state explosion* problem. It is said that a state explosion occurs when the number of sequentially reachable global states of a protocol becomes too large to be enumerated by the available computing resources.

Property 3.1 The maximum number of global states in conventional reachability analysis is provided in [Öz95] as follows:

$|\Gamma| \leq s^n \{\sum_{i=0..c} m^i\}^{n*(n-1)}$, where

- n is the number of processes in the protocol.
- s is the maximum number of states in a process.
- m is the maximum number of distinct messages sent from one process to another.
- c is the maximum number of messages in a channel, i.e., the channel capacity. □

The maximum number of global states that have to be explored grows exponentially as a function of size of the networks of CFSMs. Consider a protocol where n=5, s=10, m=5, and c=2. The maximum number of global states of the protocol according to the above formula can be $10^5*31^{20} \approx 6.72*10^{34}$. In conventional reachability analysis, all reachable global states are stored in the memory to recognize the repeated global states and therefore to guarantee the termination of the analysis. Then one may ask whether $6.72*10^{34}$ global states can be stored in the memory. Unfortunately, the answer is not positive according to the studies in [Ho90, Öz95]. Although the computing resources have significantly improved since then, it is still a major issue in reachability-based verification to provide memory efficient algorithms.

3.2 The Relief Strategies

The actual number of sequentially reachable global states of a real protocol is naturally smaller than the number calculated by Property 3.1. However, the important fact is that most of these global states are redundant for verifying the protocol against the logical errors. This fact motivated researchers to seek relief strategies for the state explosion

problem for around two decades. The following section gives an overview of some relief strategies related to the work for this thesis.

3.2.1 Fair Reachability Analysis

3.2.1.1 The Rubin-West Strategy

The first relief strategy that uses the simultaneous execution of several transitions is proposed by Rubin and West [RuWe82] for the analysis of the class of two-process protocols. The Rubin-West strategy forces the two processes to progress at the same speed during state exploration. This strategy generates and analyzes only the global states that are reached by executing equal number of transitions in each process. That is, a global state is generated from another global state by executing only one ordering of a pair of transitions, one per each process. The authors only argue that this strategy can be used to decide whether the communication is free from deadlocks. They also report a large amount of reduction in global state generation against the conventional reachability analysis. Due to that *fair progress property* (i.e., a pair of transitions formed by taking one transition per process), this strategy is later named *fair reachability analysis* by Yu and Gouda [YuGo82].

Protocol Π1 specified in Figure 2.2 of Example 2-1 is used to show some global state reduction from the Rubin-West fair reachability analysis. The *fair reachability graph* of this two-process protocol is shown in Figure 3.3. The sequentially reachable global states, which are not reachable by fair reachability analysis, are left blank in place for reference purposes. Complete conventional reachability graph of protocol Π1 is already given in Figure 3.2.

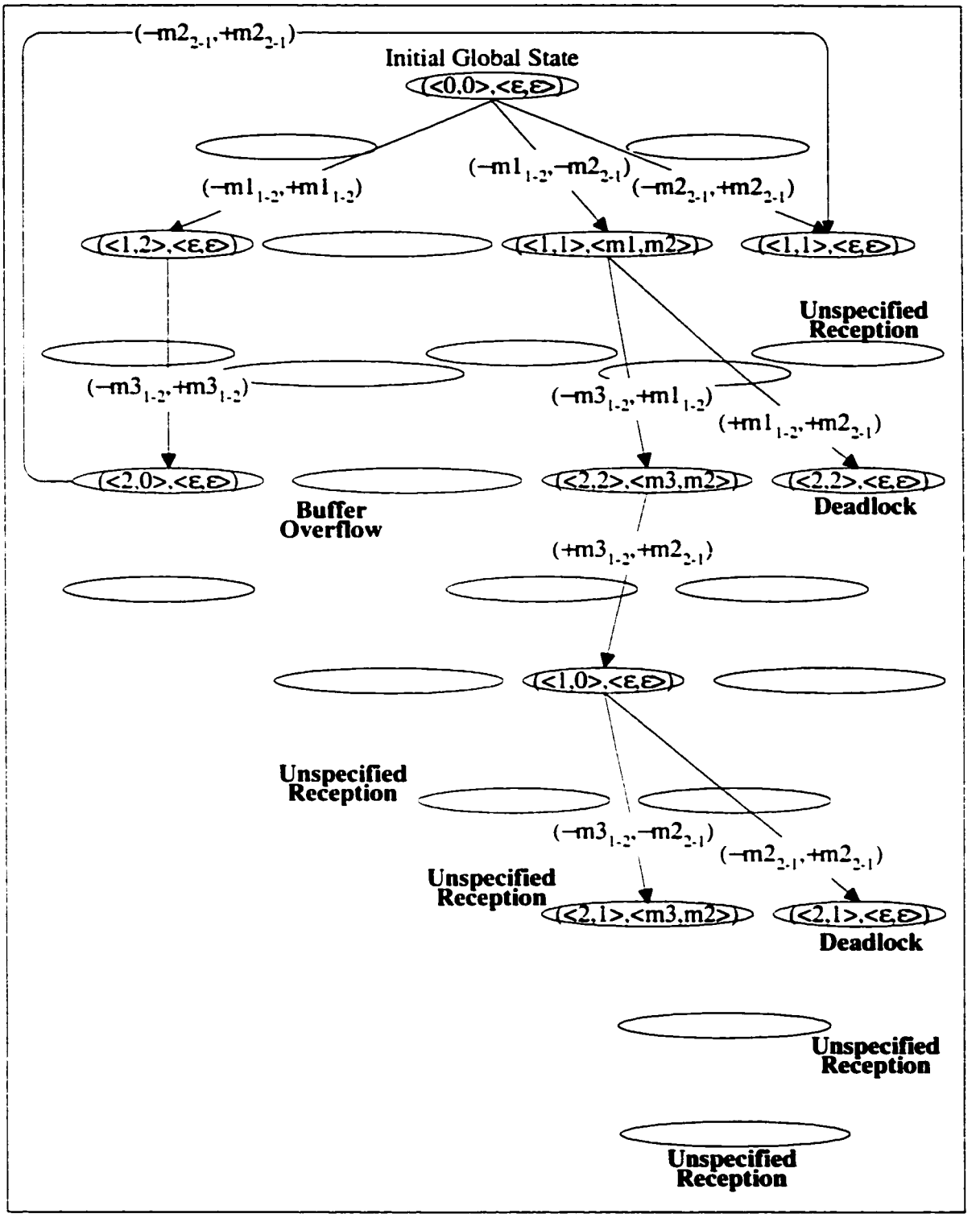

Figure 3.3 The Rubin-West fair reachability graph of two-process protocol Π1

Fair reachability analysis considers only the behavior of a protocol obtained by equal process speed of each process. Consider the simple two-process protocol in Figure 3.4. If the sender process, P_1 in this case, is faster than the receiver process, P_2 in this case, the number of messages in the channel, $C_{1,2}$ in this case, from the sender to the receiver will exceed any given bound for the channel after a finite time. However, this buffer overflow is not detected by the Rubin-West strategy as shown in the fair reachability graph of the protocol in Figure 3.5.

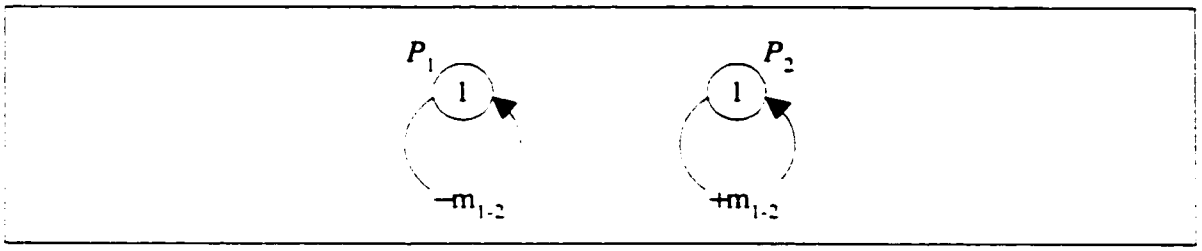

Figure 3.4 A simple protocol with a buffer overflow

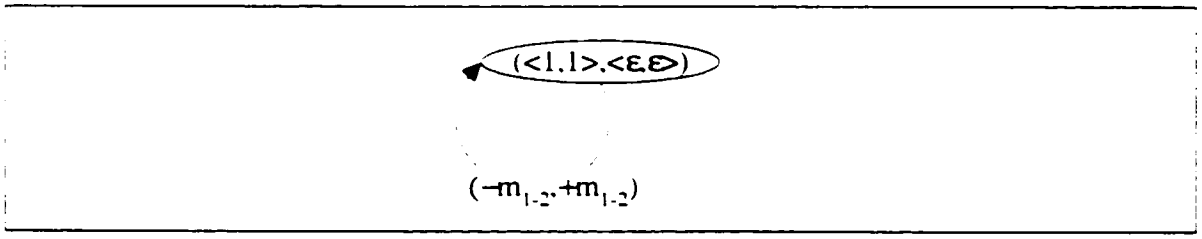

Figure 3.5 The Rubin-West fair reachability graph of the protocol in Figure 3.4

In addition, a process can include a group of transitions, which can only be executed, if one process progresses faster than the other process does. Therefore, a transition not executed during the fair reachability analysis may indeed be executable transition. The issues in the Rubin-West strategy are later cleared away by the contribution of Gouda and Han in [GoHa85].

3.2.1.2 The Yu-Gouda Strategy

Using fair reachability analysis, Yu and Gouda [YuGo82] propose a deadlock detection algorithm that is polynomial in time and space for a special two-process protocol model.

In this model, a process can send or receive one type of message and none of its states has both sending and receiving transitions. Their model is too restricted to be applicable for real communication protocols.

3.2.1.3 The Gouda-Han Strategy

Gouda and Han [GoHa85] extend the Rubin-West strategy to decide boundedness for any two-process protocol whose fair reachability graph is finite. Notice that the channels in the CFSM model used in this study are not bounded. The simple two-process protocol given in Figure 3.4 can be used as an example for the case where the fair reachability graph of a protocol is finite but the protocol is unbounded, i.e., at least one of its channel is unbounded.

[GoHa85] presents an algorithm that augments the finite reachability graph of a protocol for each process to determine the possible smallest channel capacity of each channel. For the outgoing channel from a process, the algorithm adds global states that can be reached in the fair reachability graph by only using transitions from that process.

With the contribution of Gouda and Han, fair reachability analysis can be used in its extended form to detect all logical errors in the class of two-process protocols. Many later relief strategies extend the Gouda-Han strategy, for classes of protocols having more than two processes:

- The Liu-Miller strategy generalizes fair reachability analysis for n-process cyclic protocols [LM96].
- The van der Schoot-Ural strategy generalizes fair reachability analysis for n-process multi-cyclic protocols [ScUr95b].

3.2.1.4 The Zhao-Bochmann Strategy

Zhao and Bochmann [ZhBo86] use an approach similar to fair reachability analysis but a different representation of the CFSM model. They represent protocols with process equations and reachability analysis with algebraic transformation rules. The proposed

reachability algorithm uses both sequential and fair global state transitions to generate next global states. They prove that their strategy detects deadlocks and blocking receptions that are unspecified receptions occurring only at receiving process states where all specified transitions are receptions. They also extend their strategy to detect unspecified receptions. However, the generalization of this strategy to protocols having more than two processes remains open.

3.2.1.5 The Liu-Miller Strategy

In [LiMi94a], Liu and Miller generalize fair reachability analysis for n-process protocols with a unidirectional ring topology. Due to their special topology, such protocols are called *cyclic*. Notice that any two-process protocol is a cyclic protocol, and in an n-process cyclic protocol, each process has only one input channel and one output channel. The authors observe that at every fair reachable global state for a two-process protocol, every channel has the same number of messages. They call this property the *equal channel length property*. Their generalization is based on preserving the equal channel length property for n-process cyclic protocols. Although the generalization does not preserve the equal progress property of fair reachability analysis, for protocols with n=2 the equal channel length property leads to the same reduced reachability graph as the equal progress property.

In generalized fair reachability analysis, global states are generated by executing vectors of transitions called *fair progress vectors*. A fair progress vector consists of one transition from each process. There are two types of transitions that can be included in fair progress vectors: *executable* and *enabled*. While an executable transition is defined in the conventional manner, an enabled transition is defined as follows: An enabled transition is a receiving transition that is not executable at the current global state but it is executable at least at one of the sequential immediate successors of the current global state. Actually, there is only one possible case for a receiving transition of message x is enabled at a global state, which occurs if the sender process is ready to send message x and the corresponding channel is empty at the global state. Hence, an enabled transition

in [LiMi94a] is nothing but a potentially executable receiving transition, which is defined in Definition 2.11. There are two types of fair progress vectors: *concurrency* and *synchronization*.

A concurrency vector is a transition vector, in which the transitions are either all sending or all receiving executable transitions. Notice that the execution of a concurrency vector preserves the equal channel length property because:

- The execution of a concurrency vector of sending transitions increases the length of every channel by one
- The execution of a concurrency vector of receiving transitions decreases the length of every channel by one.

A synchronization vector contains k pairs of a sending and a receiving transition such that both transitions of a pair involve the same channel, at least one sequence of transitions of a pair is executable, $k>0$, and the vector is maximal w.r.t. k. Obviously, the sending transition of a pair in a synchronization vector must be executable whereas the receiving transition of the pair can be either enabled or executable at the current global state. The execution of a synchronization vector does not change the length of any channel.

Clearly, at the initial global state and at every global state that is generated by executing a sequence of fair progress vectors from the initial global state, the equal channel length property holds.

A cyclic protocol is simultaneously unbounded if for any constant $K\geq 0$, there exists a reachable global state in which the length of every channel is greater than K, otherwise it is not simultaneously unbounded. The authors report that a given cyclic protocol has a finite fair reachability graph iff the protocol is not simultaneously unbounded. They also report that deadlock detection is decidable for the class of cyclic protocols with finite reachability graphs.

In [LiMi94a], the authors only report the results on basic formulation of their relief strategy and deadlock detection. Later, they report the results on detection of other logical errors in [LiMi94b, LiMi96].

3.2.1.6 The van der Schoot-Ural Strategy

In [ScUr95b], van der Schoot and Ural generalize fair reachability analysis for n-process protocols with a multi-cyclic topology, which is composed of m, $m \geq 1$, disjoint unidirectional rings. While each process in a cyclic protocol has only one input channel and one output channel, a multi-cyclic protocol in general allows a process to participate in more than one ring and thus, the number of input channels must be equal to the number of output channels. Each active simplex channel in a multi-cyclic protocol pertains to exactly one ring in the protocol, a requirement that implicitly confines the multiple ring topology such that, no two rings in the protocol share a common simplex channel. In addition, all simplex channels pertaining to a given ring in a multi-cyclic protocol must have equal capacity. Their generalization is based on preserving the equal channel length property for each ring, which is called *ring equilibrium property*. Recall that any two-process protocol is a cyclic protocol, hence a multi-cyclic protocol. Then, for two-process protocols, the ring equilibrium property leads to the same reduced reachability graph as the equal channel length property or the equal progress property.

In their generalized fair reachability analysis, global states are generated by executing tuples of transitions called *fair transition tuples*. There are two types of fair transition tuples: *ring tuples* and *single-channel pairs*.

Each ring tuple is associated with a ring, in which all the transitions, one from each process involved in the ring, are either sending or receiving executable transitions. Notice that the execution of a ring tuple preserves the equal channel length property in the ring because:

- The execution of a ring tuple of sending transitions increases the length of every channel in the ring by one.

- The execution of a ring tuple of receiving transitions decreases the length of every channel in the ring by one.

Each single-channel pair, which is associated with one simplex channel, consists of a sending transition and a receiving transition from any two adjacent processes such that either both are executable at the current global state, or only one of the transitions is executable while the other is potentially executable. The message correspondence must exist among the two transitions in case the receiving transition is potentially executable at the current global state. The execution of a single-channel pair does not change the length of any channel.

Clearly, at the initial global state and at every global state that is generated by executing a sequence of fair transition tuples from the initial global state, the equal channel length property holds for each ring, hence the ring equilibrium property.

Recall the definition of a simultaneously unbounded unidirectional ring in Section 3.2.1.5. A multi-cyclic protocol is simultaneously unbounded iff there exists a simultaneously unbounded ring in the protocol. The authors report that a given multi-cyclic protocol has a finite fair reachability graph iff the protocol is not simultaneously unbounded. They also report that deadlock detection is decidable for the class of multi-cyclic protocols with finite reachability graphs.

Four-process cyclic protocol Π3 specified in Figure 2.4 of Example 2-3 can be used as a demonstration for the van der Schoot-Ural strategy. A global state G_k of this protocol can be shortly represented as a pair of two quadruples: G_k=(<s_1^k, s_2^k, s_3^k, s_4^k>,<$c_{1\text{-}2}^k$, $c_{2\text{-}3}^k$, $c_{3\text{-}4}^k$, $c_{4\text{-}1}^k$>) without showing the non-active channels. This protocol has only one deadlock state that is (<2, 4, 7, 10>,<ε, ε, ε, ε>) as shown in its conventional reachability graph in Figure 3.6. Figure 3.7 shows the fair reachability graph of Π3 where the sequentially reachable global states, which are not reachable by the van der Schoot-Ural fair reachability analysis, are left blank in place for comparison purposes.

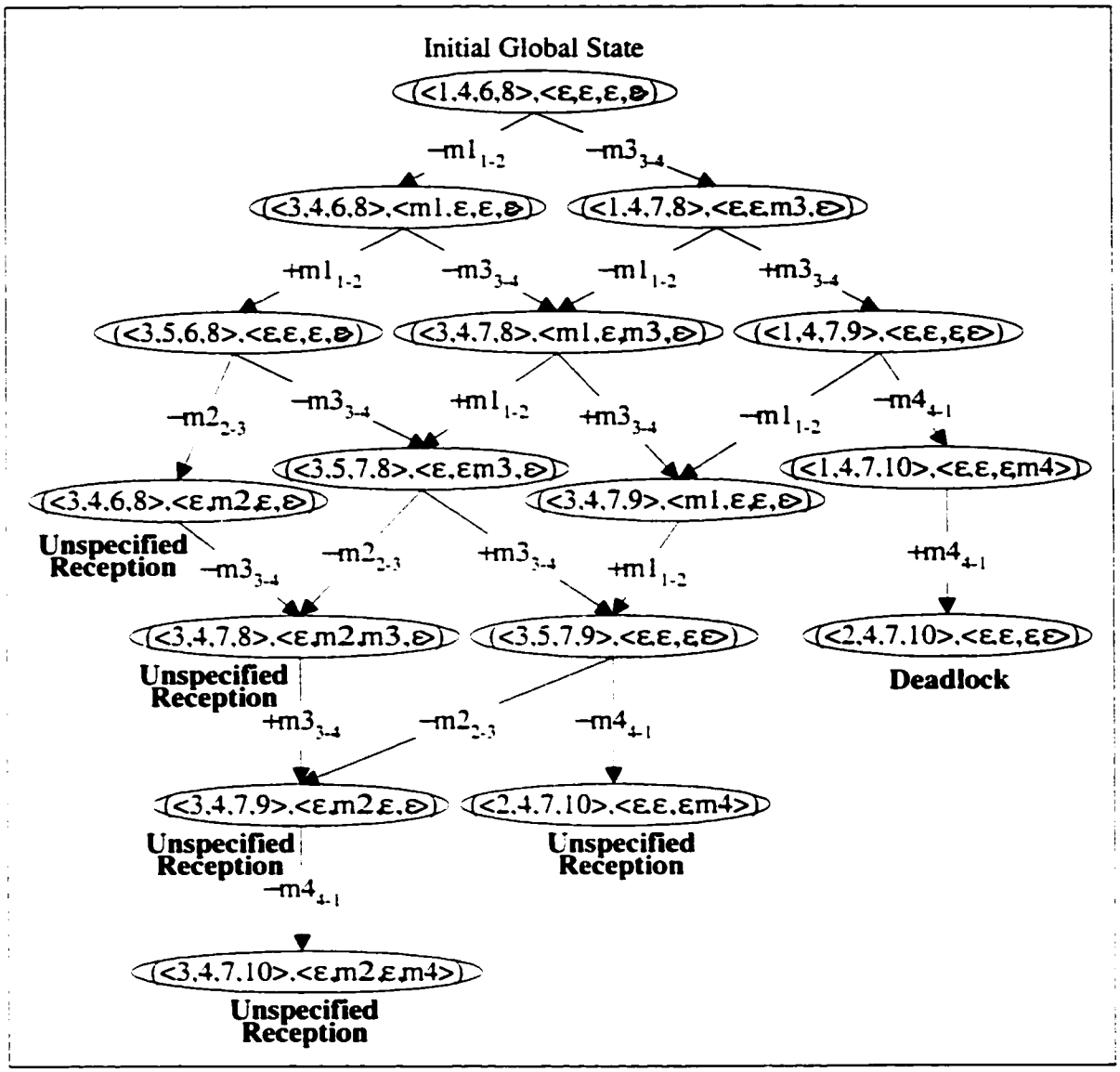

Figure 3.6 The conventional reachability graph of four-process cyclic protocol Π3

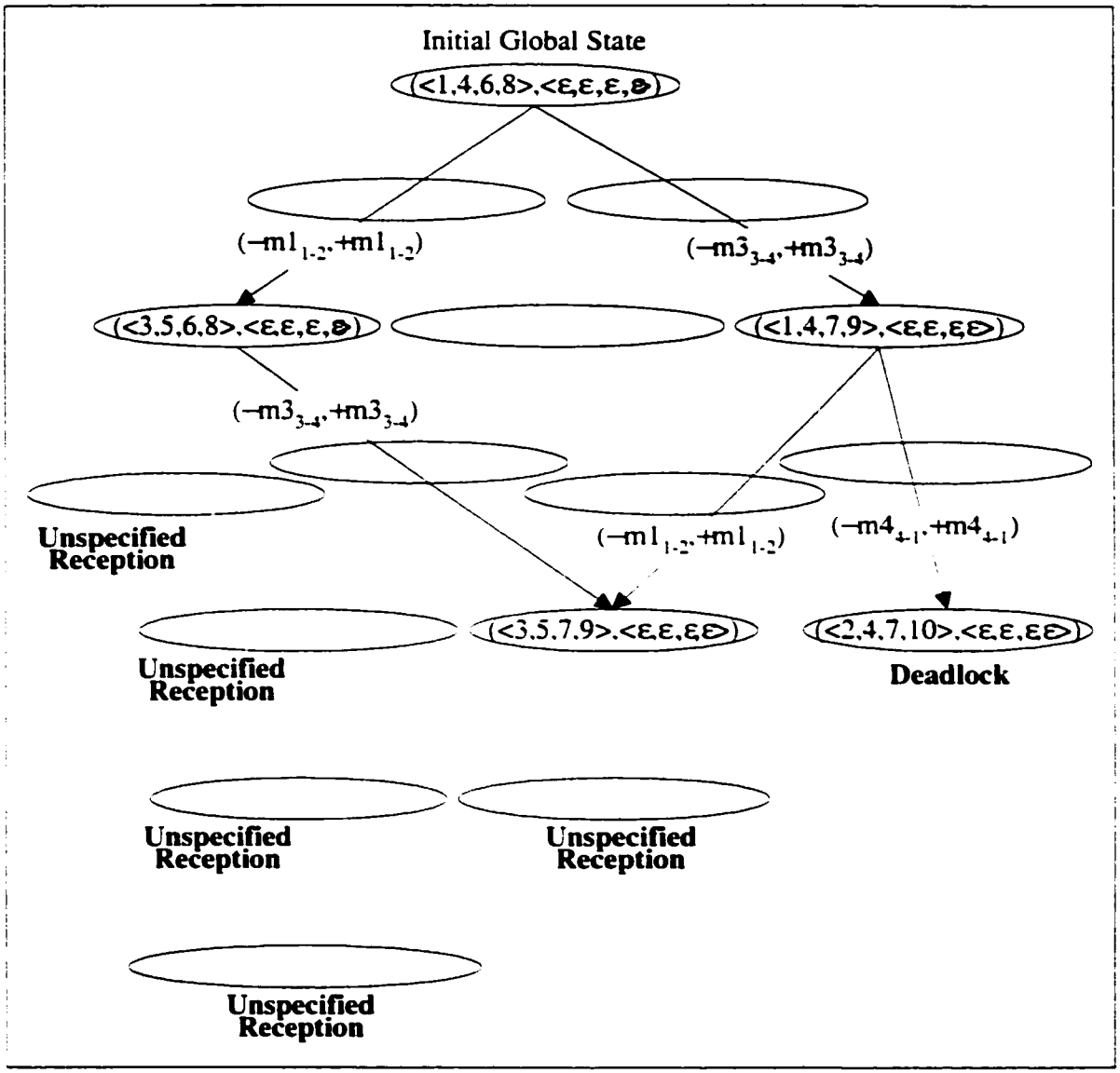

Figure 3.7 The van der Schoot-Ural fair reachability graph of cyclic protocol Π3

In [ScUr95c], van der Schoot and Ural argue that the strategy of fair reachability analysis cannot be extended beyond multi-cyclic protocols. Liu, Miller, van der Schoot, and Ural later conclude together their findings on fair reachability analysis in [LiMiScUr96].

3.2.2 Even Reachability Analysis

3.2.2.1 The Peng Strategy

In [Pe95], Peng proposes a new relief strategy, called *even reachability analysis*, for the general class of n-process protocols with arbitrary topology. The Peng strategy forces two process transitions to be executed at a time while generating global states starting from the initial global state. The generated global states then always have an even number of total length of all messages in the channels, which can be called *even total channel length property*. The author reports that the strategy is an extension of the classical fair reachability analysis [GoHan85]. However, as shown in Chapter 5, this strategy is not a generalization of fair reachability analysis. [Pe95] claims that every sequentially reachable global state with empty channels, which he calls *stable global state*, is reachable by the Peng strategy and thus, deadlock freedom of any bounded protocol is decidable. However, the Peng strategy seems not proved a full coverage of deadlock states. This strategy will be discussed at length in Chapter 5.

3.2.3 Simultaneous Reachability Analysis

3.2.3.1 The Itoh-Ichikawa Strategy

In [ItIc83], Itoh and Ichikawa extend Zafiropulo's two-process protocol model [Za78] to specify n-process protocols, and introduce so-called *reduced implementation sequences* to reduce the size of the global state space analyzed during reachability analysis. In this model, processes must synchronize on their initial states after a finite number of transitions and only those cycles passing through the initial states are allowed in each process. The Itoh-Ichikawa strategy analyzes only the behavior of the protocol between two synchronization points. Since the number of states in each process is finite, there is no embedded cycle in each process, and the processes synchronize on their initial states, the number of global states are always finite. The authors report an application of their relief strategy to a four-process protocol, which results in a much shorter computation

time and requires much less space than conventional reachability analysis. However, the protocol model limits the applicability of this strategy.

3.2.3.2 The Özdemir-Ural Strategy

In [ÖzUr95], Özdemir and Ural generalize the ideas of simultaneous execution of transitions to the general class of n-process protocols with bounded channels and without any restrictions on the topology of the protocol or the structure of the processes.

Recall that in conventional reachability analysis, the global state space is explored by the execution of a single transition in the protocol at a time. When several transitions are simultaneously executable at a global state, conventional reachability analysis considers every possible interleaving of these transitions. If there are k such transitions, then there exist $k!$ interleavings of these transitions. Even though the set of intermediate global states generated by one interleaving may not be equal to the set generated by another interleaving, these interleavings lead to a common global state. The Özdemir-Ural strategy, called *simultaneous reachability analysis*, explores only part of the global state space by generating those global states that are reachable through the simultaneous execution of transitions that are executable at a global state, and therefore prevents the generation of a large number of intermediate global states.

In simultaneous reachability analysis, the global states are generated by means of simultaneously executing selected subsets of executable transitions at a global state. Such subsets consist of at most one executable transition from each process. Özdemir and Ural call these subsets *selected simultaneous executable sets*. A selected simultaneous executable set may or may not contain an executable transition from a process having both executable and potentially executable transitions at a global state G_k, but must contain an executable transition from each process having only executable transitions at G_k. There is no restriction on the relative number of sending or receiving transitions in these subsets. The authors report that, with substantial global state and transition reduction,

- Simultaneous reachability analysis identifies every deadlock and every non-executable transition.
- By augmenting a given protocol with a very limited number of receiving transitions, simultaneous reachability analysis identifies every missing receiving transition causing an unspecified reception.
- By augmenting simultaneous reachability analysis, every overflowed channel is identified in a protocol.

3.2.3.3 The van der Schoot-Ural Strategy

In [ScUr96b], van der Schoot and Ural improve simultaneous reachability analysis by presenting *leaping reachability analysis* as a *uniform and property-driven* verification framework. Global state exploration in this framework varies quite naturally with the property to be verified, but is always based on the notion of leaping (i.e., the concurrent, or collective execution of transitions). The authors report substantial reductions in the number of global states and global transitions and thus in the required memory space and execution time, respectively. Figure 3.8 shows the LRA blocking state detection reachability graph of four-process protocol Π2, which is specified in Figure 2.3 of Example 2-2. A global state G_k of this protocol can be shortly represented as a pair of two quadruples: G_k=(<s_1^k, s_2^k, s_3^k, s_4^k>,<$c_{1,2}^k$, $c_{2,3}^k$, $c_{3,4}^k$, $c_{4,3}^k$>) without showing the non-active channels. Then, the initial state of this protocol is (<10, 20, 30, 40>,<ε, ε, ε, ε>). The number of sequentially reachable states of this protocol is forty. The LRA blocking state detection only generates two global states to verify that protocol Π2 is free from blocking states. Notice that fair reachabilty analysis is not applicable for this verification simply because Π2 is not a multi-cyclic protocol.

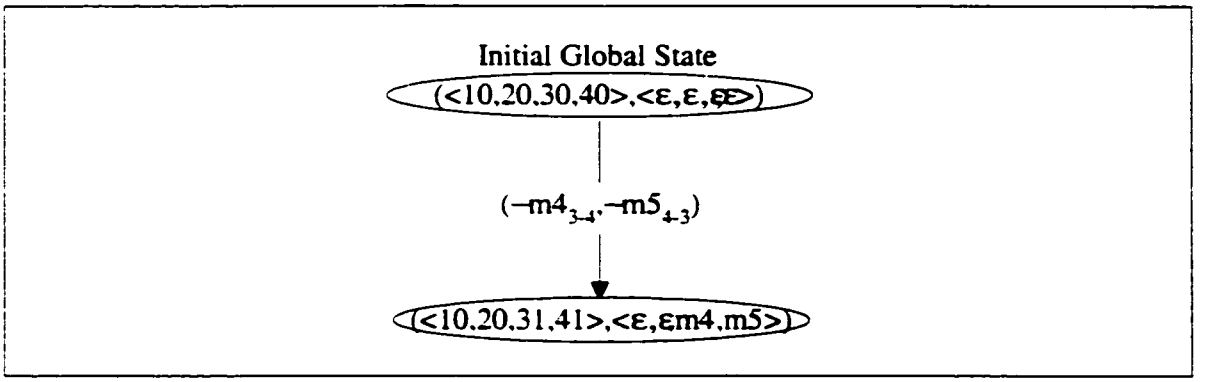

Figure 3.8 The LRA blocking state detection reachability graph of protocol Π2

3.2.4 Maximal Progress State Exploration

3.2.4.1 The Gouda-Yu Strategy

In [GoYu84], Gouda and Yu propose a relief strategy for detecting deadlocks, blocking unspecified receptions and buffer overflows in two-process protocols specified by the CFSM model, which is called *maximal progress state exploration*. The exploration is divided into two independent subtasks. In each subtask, only the global states reachable by forcing one process to make maximal progress are generated.

In conventional reachability analysis, all possible progress speeds of processes are considered. However, for two process protocols, considering only the case where one process progresses in maximal speed as in this strategy is sufficient to find deadlocks and blocking unspecified receptions. Since deadlocks and blocking unspecified receptions are the cases where a process cannot progress anymore, they can be detected by considering maximal progress of each process independently. In addition, buffer overflows are also detected by maximal progress state exploration because a buffer overflow occurs when one process transmits messages faster than the other process consumes them. Maximal progress state exploration may not be as efficient as fair reachability analysis in the Rubin-West strategy [RuWe82], but it detects also buffer overflows and its independent subtasks can be simultaneously executed to reduce the execution time. The generalization of this strategy to protocols consisting of more than two processes remains open.

The CFSM model lends itself to reachability analysis, a global state space exploration technique that has been advocated for the verification of the absence of logical errors in a protocol. In reachability analysis, all possible reachable global states of a protocol are generated, starting from the initial global state and allowing only one process to progress at a time, while checking each of these states for the logical errors. Two well-known limitations to the reachability analysis technique are undecidability and state explosion. One subclass of protocols for which the verification problem is decidable consists of many practical protocols where all of the channels are bounded. However, even with bounded channels, protocols often have a very large global state space and hence exhaustive generation of the global state space by reachability analysis gives rise to the state explosion problem. Many strategies have been proposed to relieve the state explosion problems. Besides the relief strategies mentioned in this chapter, there are other relief strategies such as *Duologue-matrix analysis* [Za78], *acyclic expansion of protocols* [BrZa83], *random-walk state exploration* [We86], *decomposition of structured protocols* [VuCo82], *partial-order verification based on ample sets* [ScUr98b]. Further information can be found in the survey papers [LiChLi87, Yu88].

Chapter 4 will present the RELIEF tool that implements the conventional reachability analysis technique (CRA) along with some relief strategies such as fair reachability analysis (FRA), simultaneous reachability analysis (SRA), leaping reachability analysis (LRA), even reachability analysis (ERA), partial-order verification based on ample sets (POVAS) to compare their performances with CRA in terms of number of generated global states and global transitions.

Chapter 4 The RELIEF Tool

The initial version of RELIEF was developed in the summer of 1994 by Melih Özdemir and Kadir Özdemir in forms of working programs using C under Unix to support the research on simultaneous reachability analysis (SRA). Since then, Tuong Nguyen has been developing it to be a Unix-based research package tool while extending it to include various strategies of reachability analysis such as fair reachabilty analysis (FRA), leaping reachability analysis (LRA), partial-order verification based on ample sets (POVAS), and recently even reachability analysis (ERA). The work is under the supervision of Professor Ural. The title page of the current version of RELIEF is captured in Figure 4.1.

```
{site0.site.uottawa.ca} relief
  ************************************************************************

                          -- RELIEF Version 4.0 --
                                 (1994-2002)
                                Tuong Nguyen
                           Melih Ozdemir, Kadir Ozdemir
                                  Hasan Ural

                                    TSERG
            (Telecommunications Software Engineering Research Group)
                         Department of Computer Science
                             University of Ottawa
                        Ottawa, Ontario, Canada, K1N 6N5

                                  Sponsored by
                                     TRIO
               (Telecommunications Research Institute of Ontario)
                                    NSERC
          (Natural Sciences and Engineering Research Council of Canada)

  ************************************************************************

Press Return to continue...
```

Figure 4.1 The RELIEF title page

This chapter presents the software architecture of the RELIEF tool, introduces the input and output formats used in RELIEF, and shows how the empirical studies on the relief strategies are carried out using the tool. The sections in this chapter are then arranged as follows: Section 4.1 gives an overview of the RELIEF software architecture and Section 4.2 describes the empirical study using the RELIEF tool.

4.1 The RELIEF Software Architecture

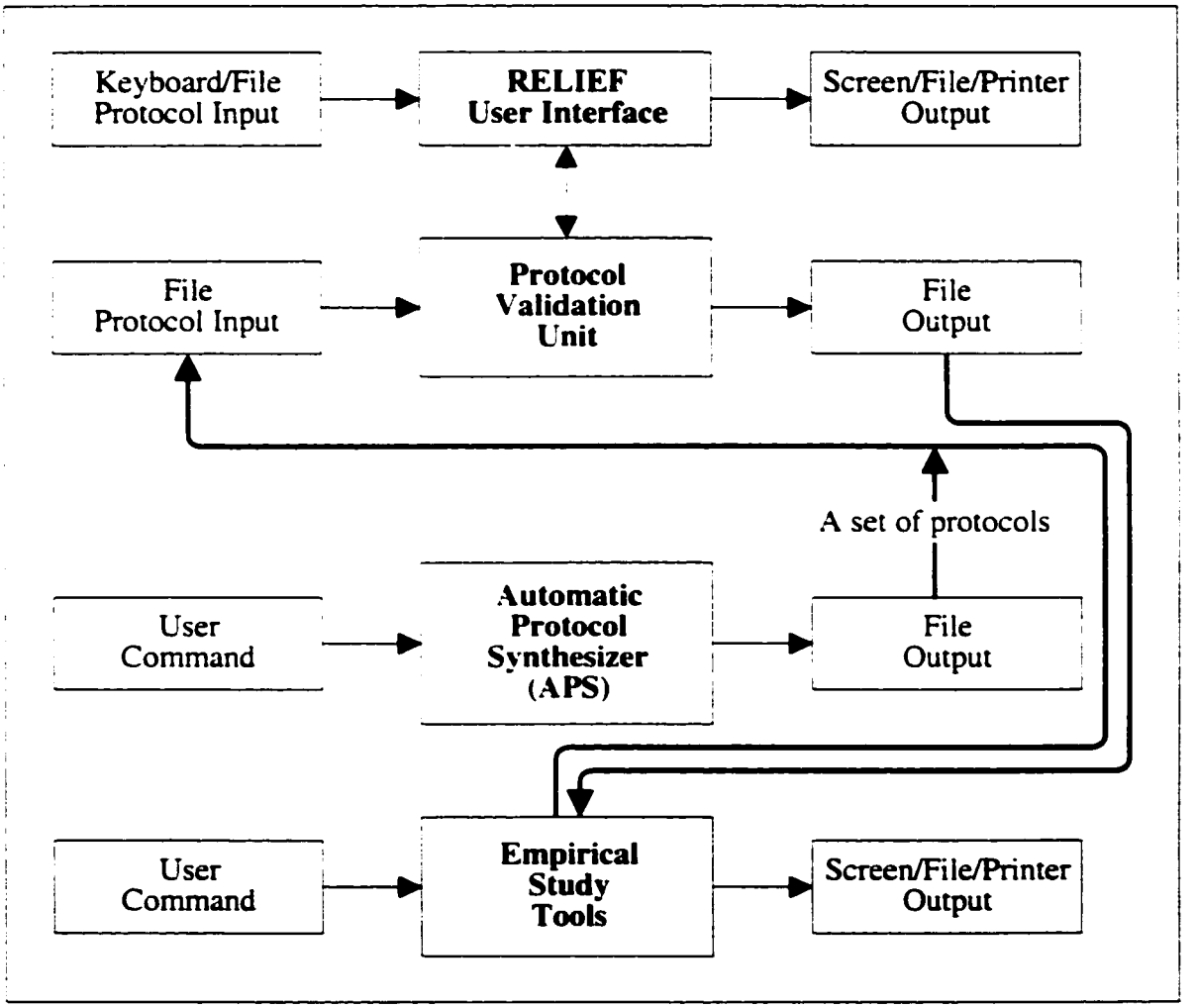

Figure 4.2 The RELIEF architecture overview

Overview of the RELIEF software architecture is provided in Figure 4.2. There are four main components:

- The user interface.
- The protocol validation unit.
- The *automatic protocol synthesizer.*
- The empirical study tools.

The protocol validation unit accepts either a *single-protocol* or a *multi-protocol* input file to validate by a particular relief strategy, and produces as output a validation file recording the logical errors detected by the strategy. The user interface of RELIEF is text-based and menu-driven, which assists users to formulate a single-protocol input file via either a system editor or an interactive editor specially developed for RELIEF. Users then can select the relief strategy from a menu and the user interface will interact with the protocol validation unit to get the validation output file. Finally, the user interface will interpret the results in an intuitive form and display it to users. The automatic protocol synthesizer (APS) is a stand-alone component that synthesizes a large number of protocols as desired by users and store them as a multi-protocol input file. The empirical study tools pick up that multi-protocol input file and repeatedly send it to the protocol validation unit to get the validation output files either by CRA or by various relief strategies. Upon receiving enough validation results, the empirical study tools compare and calculate the reduction percentage in generated global states, global transitions, used memory space, and execution time. The empirical study tools can also analyze a set of APS protocols to produce as output the statistical properties of the protocols in the set. Users always have choice to read, save, or file a report from the empirical study tools.

4.1.1 The User Interface

The menu hierarchy of RELIEF starting from the main menu is illustrated in Figure 4.3.

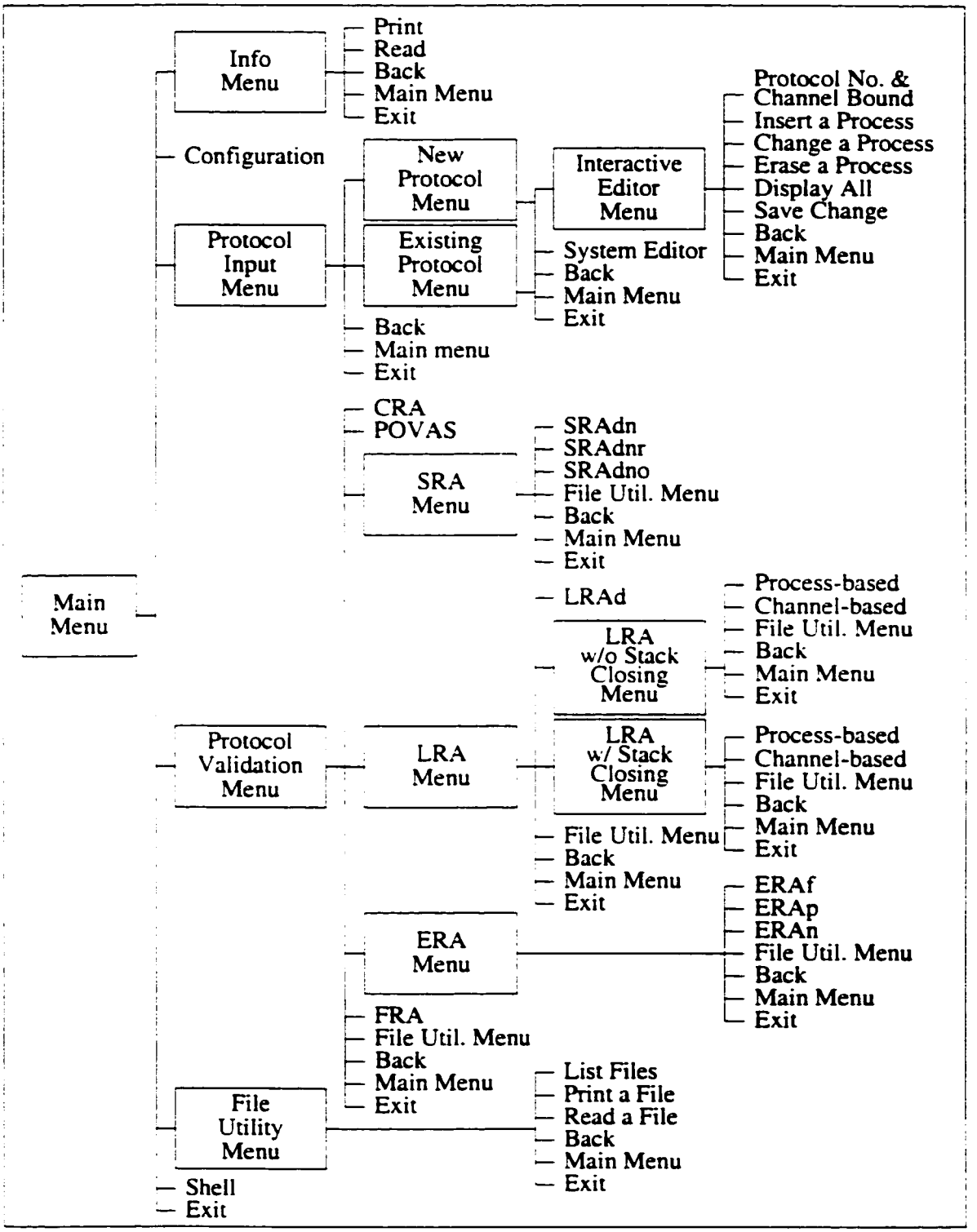

Figure 4.3 The RELIEF menu hierarchy

4.1.1.1 Main Menu and The Configuration Item

The RELIEF main menu is captured in Figure 4.4. At the first time of running RELIEF, users need to select the option to configure the file path that enables users to read the information files about the tool. Users can also indicate their favorite system editor if they decide to formulate the single-protocol input file using a system editor instead of the interactive editor specially developed for RELIEF. The default system editor is 'vi'. Users can also decide the print command and the destination printer as they prefer for later use within RELIEF. The configuration information is automatically saved in the Unix hidden file '.reliefrc' that is stored in the users' home directory. Users do not need to configure again in subsequent running of RELIEF unless they change their mind.

```
*************************************************************************
                              MAIN MENU

        Information.....................................................(i)
        Configuration...................................................(c)
        Protocol Input..................................................(p)
        Protocol Validation.............................................(v)

        File Utilities..................................................(u)
        Shell...........................................................(s)
        Exit............................................................(x)

        Choice..........................................................(c

Current File Path:  /a/infoaf/usr3/grad/tunguyen/Bin/
Enter new File Path (Return for no change):

Current System Editor: vi
Enter new System Editor (Return for no change):

Current Print Command: mpage -2
Enter new Print Command (Return for nc change):

Current Printer Name: prgps2
Enter new Printer Name (Return for no change): assertps1

Press Return to continue...
```

Figure 4.4 The configuration menu item

4.1.1.2 The Protocol Input Menu

Recall that the protocol validation unit accepts only protocol input as a file. From the main menu, users can use the RELIEF user interface to formulate the protocol input file

using either their favorite system editor or the interactive editor provided with RELIEF. An example of using the protocol input menu is captured in Figure 4.5.

```
**************************************************************************
                          PROTOCOL INPUT MENU

        New Protocol.....................................................(n)
        Existing Protocol................................................(e)

        Back to Previous Menu............................................(b)
        Main Menu........................................................(m)
        Exit.............................................................(x)

        Choice...........................................................(●

Enter protocol file name: protocol1

**************************************************************************
                              EDITOR MENU

        Interactive Editor...............................................(i)
        System Editor....................................................(s)

        Back to Previous Menu............................................(b)
        Main Menu........................................................(m)
        Exit.............................................................(x)

        Choice...........................................................(
```

Figure 4.5 The protocol input menu

Novices to RELIEF are suggested to use the interactive editor that step-by-step assists users to enter the protocol. Once being familiar with RELIEF, using a system editor may be faster. The user interface displays the protocol to users in an intuitive form that is called the *external presentation* of the protocol as in Figure 4.6. Actually, RELIEF uses the protocol information saved in the *internal file* whose format is less intuitive. Figure 4.7 shows part of the internal file that stores the information of the same protocol in Figure 4.6. The format of a protocol in the internal file is shown in Figure 4.8.

```
Protocol No.: 1
Channel Bound: 2
Number of Processes: 2

Process No.: 1
States Nos.: 0 1 2 3
*****************************************************************************
      State          Message          Type        Channel       Next State
*****************************************************************************
      0              m1               -           2             1
      0              m2               +           2             1
      1              m3               -           2             2
      1              m2               +           2             2
      1              m4               +           2             3
      2              m2               +           2             1
      3              m1               +           2             0

Process No.: 2
States Nos.: 0 1 2 3
*****************************************************************************
      State          Message          Type        Channel       Next State
*****************************************************************************
      0              m2               -           1             1
      0              m1               +           1             2
      1              m1               +           1             2
      2              m1               +           1             3
      2              m3               +           1             0
      3              m4               -           1             2
```

Figure 4.6 The external presentation for two-process protocol Π1

```
1
2 1 2
4 0 1 2 3
2
m1 - 2 1
m2 + 2 1
3
m3 - 2 2
m2 + 2 2
m4 + 2 3
1
m2 + 2 1
1
m1 + 2 0
4 0 1 2 3
2
m2 - 1 1
m1 + 1 2
1
m1 + 1 2
2
m1 + 1 3
...
```

Figure 4.7 Part of the internal file representing two-process protocol Π1

```
Protocol ::=
    <Protocol_No.>
    <Number_of_Processes> <Process_No.> <Process_No.> {<Process_No.>}
    <FSM>
    {<FSM>}
FSM ::=
    <Number_of_States> Initial_State {<State_No.>}
    <Number_of_Transitions>
    {<Transition>}
Transition ::=
        <Message> <Type> <Channel> <Next_State>
```

Figure 4.8 The format of a protocol in the internal file

4.1.1.3 The Protocol Validation Menu

Once the single-protocol input file is ready, users can move to the protocol validation menu as in Figure 4.9. CRA and a number of relief strategies are available. Some of them are action items while the others lead to submenus (Figure 4.3).

```
**************************************************************************
                        PROTOCOL VALIDATION MENU

      CRA   - Conventional Reachability Analysis..............(c)
      POVAS - Partial Order Validation based on Ample Sets.....(p)
      SRA   - Simultaneous Reachability Analysis..............(s)
      LRA   - Leaping Reachability Analysis...................(l)
      ERA   - Even Reachability Analysis......................(e)
      FRA   - Fair Reachability Analysis......................(f)

      File Utilities..........................................(u)
      Back to Previous Menu...................................(b)
      Main Menu...............................................(m)
      Exit....................................................(x)

      Choice..................................................(
```

Figure 4.9 The protocol validation menu

```
CONVENTIONAL REACHABILITY ANALYSIS (CRA)
Logical Error Detector reports:

PROTOCOL No.: 1
NUMBER OF PROCESSES: 2
Process No. 1: States Nos.: 0 1 2 3
Process No. 2: States Nos.: 0 1 2 3
NUMBER OF  CHANNELS: 2
Channel Bound: 2
Channel  Nos.: 1-2 2-1

  BLOCKING STATES:
  ********************************************************************
   <State i,State j,...>,<Queue i-j,...,Queue j-i,...>
  ********************************************************************
   <2,1>,<,>
   <2,1>,<m3,>
   <2,1>,<m3.m3,>
   <2,2>,<,>

  UNSPECIFIED RECEPTION CAUSING (PROCESS STATE,MESSAGE) PAIRS:
  ********************************************************************
          Process No.       State             Message      (from) Process No.
  ********************************************************************
          2                 0                 m3                  1
          2                 1                 m3                  1

  BUFFER OVERFLOW CAUSING (PROCESS STATE,MESSAGE) PAIRS:
  ********************************************************************
          Process No.       State             Message        (to) Process No.
  ********************************************************************
          1                 1                 m3                  2

  NON-EXECUTABLE TRANSITIONS:
  ********************************************************************
       Process No.  State      Message     Type      Channel     Next State
  ********************************************************************
       1            1          m4          +         2           3
       1            3          m1          +         2           0
       2            2          m1          +         1           3
       2            3          m4          -         1           2

  NUMBER OF PROTOCOL DESIGN ERRORS:
  ********************************************************************
      Blocking     Deadlock     UR Causing     BO Causing     Non-Executable
      States       States       (process,M)    (process,M)    Transitions
                                Pairs          Pairs
  ********************************************************************
      4            2            2              1              4

  Global States      : 28
  Global Transitions: 38
  Space Consumed     : 1.39 (KBytes)
  Time Consumed      : 0.00 (Seconds)

Save report (y/n)? n

Print report (y/n)? n

Press Return to continue...
```

Figure 4.10 The CRA validation results of two-process protocol Π1

```
FAIR REACHABILITY ANALYSIS (FRA)
Deadlock Detector reports:

PROTOCOL No.: 1
NUMBER OF PROCESSES: 2
Process No. 1: States Nos.: 0 1 2 3
Process No. 2: States Nos.: 0 1 2 3
NUMBER OF  CHANNELS: 2
Channel Bound: 2
Channel  Nos.: 1-2 2-1

  BLOCKING STATES:
  ********************************************************************
   <State i,State j,...>,<Queue i-j,...,Queue j-i,...>
  ********************************************************************
   <2,1>,<,>
   <2,2>,<,>

  UNSPECIFIED RECEPTION CAUSING (PROCESS STATE,MESSAGE) PAIRS:
  ********************************************************************
          Process No.      State            Message      (from) Process No.
  ********************************************************************
          2                1                m3                  1

  NUMBER OF PROTOCOL DESIGN ERRORS:
  ********************************************************************
      Blocking     Deadlock     UR Causing     BO Causing     Non-Executable
      States       States       (process,M)    (process,M)    Transitions
                                Pairs          Pairs
  ********************************************************************
      >= 2         2            >= 1           >= 0           ?

  Global States      : 10
  Global Transitions: 10
  Space Consumed     : 0.50 (KBytes)
  Time Consumed      : 0.02 (Seconds)

Save report (y/n)? n

Print report (y/n)? n

Press Return to continue...
```

Figure 4.11 The FRA validation results of two-process protocol Π1

Recall two-process protocol Π1 specified in Figure 2.2 of Example 2-1, the conventional reachability and the fair reachability graphs of this protocol are given in Figure 3.2 and Figure 3.3, respectively. Now, the CRA and FRA validation results by RELIEF are captured in Figure 4.10 and Figure 4.11, respectively. While CRA can detect every type of logical errors, FRA can detect only deadlocks but with a large reduction in the number of generated global states and global transitions as shown. Notice that a period, ., is used as the separator for the messages in a queue, a dash, -, is used as the

separator for the process indices when specifying a channel, and an empty channel is left empty because the symbol ε of the empty sequence is not available in the tool.

```
CONVENTIONAL REACHABILITY ANALYSIS (CRA)
Logical Error Detector reports:

PROTOCOL No.: 2
NUMBER OF PROCESSES: 4
Process No. 1: States Nos.: 10 11 12
Process No. 2: States Nos.: 20 21 22
Process No. 3: States Nos.: 30 31
Process No. 4: States Nos.: 40 41
NUMBER OF  CHANNELS: 4
Channel Bound: 5
Channel  Nos.: 1-2 2-3 3-4 4-3

  UNSPECIFIED RECEPTION CAUSING (PROCESS STATE,MESSAGE) PAIRS:
  ************************************************************
         Process No.      State            Message     (from) Process No.
  ************************************************************
         2                21               m2                  1
         3                30               m5                  4
         3                30               m3                  2
         3                31               m3                  2
         4                40               m4                  3

  NON-EXECUTABLE TRANSITIONS:
  ************************************************************
       Process No.  State      Message     Type      Channel     Next State
  ************************************************************
       1            10         m1          -         4           12

  NUMBER OF PROTOCOL DESIGN ERRORS:
  ************************************************************
      Blocking     Deadlock     UR Causing     BO Causing    Non-Executable
      States       States       (process,M)    (process,M)   Transitions
                                Pairs          Pairs
  ************************************************************
      0            0            5              0             1

  Global States      : 40
  Global Transitions: 100
  Space Consumed     : 2.04 (KBytes)
  Time Consumed      : 0.02 (Seconds)

Save report (y/n)? n

Print report (y/n)? n

Press Return to continue...
```

Figure 4.12 The CRA validation results of four-process protocol Π2

Figure 4.12 captures the CRA validation results of four-process protocol Π2 specified in Figure 2.3 of Example 2-2. This protocol is not multi-cyclic as checked by

RELIEF in Figure 4.13. Protocol Π2 fails at the first condition in the definition of multi-cyclic protocols (Definition 2.5): its communication topology graph is not strongly connected as a whole but scattered as three separate components. Therefore, FRA is not applicable in this case.

```
**************************************************************************
                        PROTOCOL VALIDATION MENU

        CRA   - Conventional Reachability Analysis..............(c)
        POVAS - Partial Order Validation based on Ample Sets.....(p)
        SRA   - Simultaneous Reachability Analysis..............(s)
        LRA   - Leaping Reachability Analysis...................(l)
        ERA   - Even Reachability Analysis......................(e)
        FRA   - Fair Reachability Analysis......................(f)

        File Utilities..........................................(u)
        Back to Previous Menu...................................(b)
        Main Menu...............................................(m)
        Exit....................................................(x)

        Choice..................................................(f

Not Strongly-Connected! Cannot apply FRA to Protocol No. 2

Read report (y/n)? y


Protocol No.: 2
Process Nos.: 1 2 3 4

  STRONGLY CONNECTED COMPONENTS:
  **************************************************************************
                           Component No.       Process No.
  **************************************************************************
                           1                   4
                           1                   3

                           2                   2

                           3                   1

Save report (y/n)? n

Print report (y/n)? n
```

Figure 4.13 The multi-cyclic check for four-process protocol Π2

However, leaping reachability analysis (LRA) can be used for this protocol because LRA is not restricted by any topology. The LRA blocking state detection results of protocol Π2 by using RELIEF is given in Figure 4.14. From those results, LRA needs to explore only two global states to verify that the protocol is free from blocking states,

which include deadlocks, while CRA can verify it only after exploring forty global states. The same observation can be found in Figure 3.8 of Section 3.2.3.3 when leaping reachability analysis is first mentioned in this thesis.

```
LEAPING REACHABILITY ANALYSIS (LRAd)
Deadlock and Blocking State Detector reports:

PROTOCOL No.: 2
NUMBER OF PROCESSES: 4
Process No. 1: States Nos.: 10 11 12
Process No. 2: States Nos.: 20 21 22
Process No. 3: States Nos.: 30 31
Process No. 4: States Nos.: 40 41
NUMBER OF  CHANNELS: 4
Channel Bound: 5
Channel  Nos.: 1-2 2-3 3-4 4-3

  NUMBER OF PROTOCOL DESIGN ERRORS:
  ********************************************************************
      Blocking      Deadlock      UR Causing      BO Causing     Non-Executable
      States        States        (process,M)     (process,M)    Transitions
                                  Pairs           Pairs
  ********************************************************************
      0             0             >= 0            >= 0             ?

  Global States      : 2
  Global Transitions: 2
  Space Consumed     : 0.12 (KBytes)
  Time Consumed      : 0.00 (Seconds)

Save report (y/n)? n

Print report (y/n)? n

Press Return to continue...
```

Figure 4.14 The LRA blocking state detection results of four-process protocol Π2

The next protocol in the examples given in Section 2.2 is four-process cyclic protocol Π3 specified in Figure 2.4 of Example 2-3. Recall that protocol Π3 has been used as a demonstration for the van der Schoot-Ural fair reachability analysis (Section 3.2.1.6). The CRA and FRA validation results of protocol Π3 by the RELIEF tool are captured in Figure 4.15 and Figure 4.16, respectively.

```
CONVENTIONAL REACHABILITY ANALYSIS (CRA)
Logical Error Detector reports:

PROTOCOL No.: 3
NUMBER OF PROCESSES: 4
Process No. 1: States Nos.: 1 2 3
Process No. 2: States Nos.: 4 5
Process No. 3: States Nos.: 6 7
Process No. 4: States Nos.: 8 9 10
NUMBER OF  CHANNELS: 4
Channel Bound: 5
Channel  Nos.: 1-2 2-3 3-4 4-1

  BLOCKING STATES:
  ********************************************************************
   <State i,State j,...>,<Queue i-j,...,Queue j-i,...>
  ********************************************************************
   <2,4,7,10>,<,,,,,,,,,,>
   <3,4,7,10>,<,,,m2,,,,,m4,,>

  UNSPECIFIED RECEPTION CAUSING (PROCESS STATE,MESSAGE) PAIRS:
  ********************************************************************
          Process No.      State            Message     (from) Process No.
  ********************************************************************
          1                3                m4                 4
          3                6                m2                 2
          3                7                m2                 2

  NUMBER OF PROTOCOL DESIGN ERRORS:
  ********************************************************************
      Blocking     Deadlock     UR Causing     BO Causing    Non-Executable
      States       States       (process,M)    (process,M)   Transitions
                                Pairs          Pairs
  ********************************************************************
      2            1            3              0               0

  Global States      : 17
  Global Transitions: 25
  Space Consumed     : 0.85 (KBytes)
  Time Consumed      : 0.00 (Seconds)

Save report (y/n)? n

Print report (y/n)? n

Press Return to continue...
```

Figure 4.15 The CRA validation results of four-process cyclic protocol Π3

```
FAIR REACHABILITY ANALYSIS (FRA)
Deadlock Detector reports:

PROTOCOL No.: 3
NUMBER OF PROCESSES: 4
Process No. 1: States Nos.: 1 2 3
Process No. 2: States Nos.: 4 5
Process No. 3: Scates Nos.: 6 7
Process No. 4: States Nos.: 8 9 10
NUMBER OF  CHANNELS: 4
Channel Bound: 5
Channel  Nos.: 1-2 2-3 3-4 4-1

  BLOCKING STATES:
  ********************************************************************
   <State i,State j,...>,<Queue i-j,...,Queue j-i,...>
  ********************************************************************
   <2,4,7,10>,<,,,,,,,,,,>

  NUMBER OF PROTOCOL DESIGN ERRORS:
  ********************************************************************
      Blocking      Deadlock      UR Causing      BO Causing     Non-Executable
      States        States        (process,M)     (process,M)    Transitions
                                  Pairs           Pairs
  ********************************************************************
      >= 1          1             >= 0            >= 0             ?

  Global States      : 5
  Global Transitions: 5
  Space Consumed     : 0.25 (KBytes)
  Time Consumed      : 0.02 (Seconds)

Save report (y/n)? n

Print report (y/n)? n

Press Return to continue...
```

Figure 4.16 The FRA validation results of four-process cyclic protocol Π3

The last protocol in the examples given in Section 2.2 is four-process multi-cyclic protocol Π4, which is specified in Figure 2.5 of Example 2-4. However, RELIEF cannot directly validate this protocol because in RELIEF, the channel bound must be fixed for every channel in the protocol, while protocol Π4 has two different channel bounds for two unidirectional rings. Actually, it is not an issue because choosing a common capacity for every channel is normally possible in practice.

4.1.2 The Protocol Validation Unit

A list of relief strategies that are currently supported in RELIEF can be viewed from the protocol validation menu in Figure 4.3. The most important things in the implementation of reachability analysis are the data structures to store global states and reachability graph in RAM. No matter how many relief strategies are implemented in RELIEF, they should share those common data structures and the procedures to operate them. Some of such data structures and procedures are presented in this section.

4.1.2.1 The Global State Data Structure

As defined in the CFSM model (Definition 2.1), a global state consists of an array of process states, one state for each process, and the information about the content of each channel. Array *states* of size n=MAX_PROCESSES is used to store the process states. For the channel contents, instead of allocating memory space for a multi-dimensional array to store $n(n-1)$ message queues of channel-bound length, the data structures use only one string to store all non-empty message queues. The string is dynamic memory allocated, then pointer *queueP* is needed to hold the string of all non-empty message queues. Variable *queuLength* stores the length of the string pointed by *queueP*. If all channels are empty, then *queuLength*=0 and *queueP*=NULL. Array *channelBitMap* of size MAX_CBYTES that must be eventually equivalent to at least $n(n\text{-}1)$ bits is to identify the non-empty channels.

```
typedef struct globalStateStruct {
    char states[MAX_PROCESSES];          /* Process states */
    char channelBitMap[MAX_CBYTES];  /* Channel bit map */
    int  queueLength;  /* Size in bytes of the string pointed by queueP */
    char *queueP;      /* Pointer to string of all non-empty message queues */
} globalStateType, *globalStatePType;
```

Figure 4.17 The global state structure in RELIEF

The structure model is introduced by Kadir Özdemir. It is very efficient in terms of memory space saving but less straightforward in terms of usage. Hence, an additional data structure is used to manage the channel information, so called the channel manager. The channel manager uses two-dimensional array *first* to locate the head of each message queue in the string. The next messages in that queue are stored contiguously after the head message up to the *size* of the queue that is also provided in the channel manager structure. In addition, the channel manager stores an updated total count of non-empty channels for fast access.

```
typedef struct channelManStruct {
    char num;       /* number of non-empty channels */
    char size[MAX_PROCESSES][MAX_PROCESSES];
    int first[MAX_PROCESSES][MAX_PROCESSES];
} channelManType, *channelManPType;
Public function:
CU_ManageAllChannels(const   globalStatePType globalStateP,
                             channelManPType channelManP);
```

Figure 4.18 The channel manager structure in RELIEF

Common utility function *CU_ManageAllChannels*() retrieves the channel information from the global state structure pointed by the first argument and arranges the information into the channel manager structure pointed by the second argument. In Figure 4.19, the global state structure is exemplified by a function to print out the content of the channel from *sender* to *receiver*.

```
void CU_PrintOneMessageQueue(const globalStatePType globalSP,
                             processIndexType sender,
                             processIndexType receiver)
{
    channelManPType    cManP;    /* Pointer to a channel manager struct */
    int                size;     /* Size of the message queue */

    CU_ManageAllChannels(globalSP, cManP);
    size=cManP→size[sender][receiver];
    if (!size) {
        printf("Channel %d-%d is empty\n", sender, receiver);
    } else {
        char*  stringP;   /* Pointer to a string of characters */
        int i;            /* Loop index */

        /* Make stringP point to the head message of the channel queue */
        stringP=(globalSP→queueP)+(cManP→first[sender][receiver]);
        printf("Content of channel %d-%d: ", sender, receiver);
        for (i=0; i<size; i++) {
            printf("%s", stringP+i);
            if (i==size-1) printf("\n");
            else printf(".");
        }
    }
}
```

Figure 4.19 The RELIEF function to print out a global state

Notice that, this function is modified from the original version to fit in the context of this thesis. In fact, both processes and messages are stored by their indices in the FIFO queues. When printing out, they need to be converted to the external presentation format using the mapping functions.

4.1.2.2 The Generic Procedure for Reachability Analysis

Section 3.1 proposes an algorithm for conventional reachability analysis. In fact, the algorithm can be adapted not only for conventional reachability analysis but also for all relief strategies in the protocol validation unit of RELIEF. Each of them explores the current global state and applies their own strategy to produce none or several next global states.

If the next global states have not been explored yet, they are pushed one by one onto an internal stack structure and inserted into an internal tree structure. Then the global state on the top of the stack will be taken as the current global state and the exploration process will repeat until the stack is empty. Even being removed from the stack, the global state remains in the storage pool for the search when a next global state is produced. The generic procedure for reachability analysis is depicted in Figure 4.20. The search in the storage pool to find out whether a global state has been explored is facilitated by an internal structure that is a BLACK/RED search tree [CoLeRi90].

According to Section 3.1, when a stack structure is used, the global states in the reachability graph are explored in order of a depth-first search. Figure 4.21 captures the order in which the sequential global states are explored by RELIEF. Recall that the conventional reachability graph of $\Pi 1$ is already given in Figure 3.2, which can be used along with Figure 4.21 to exemplify the depth-first search.

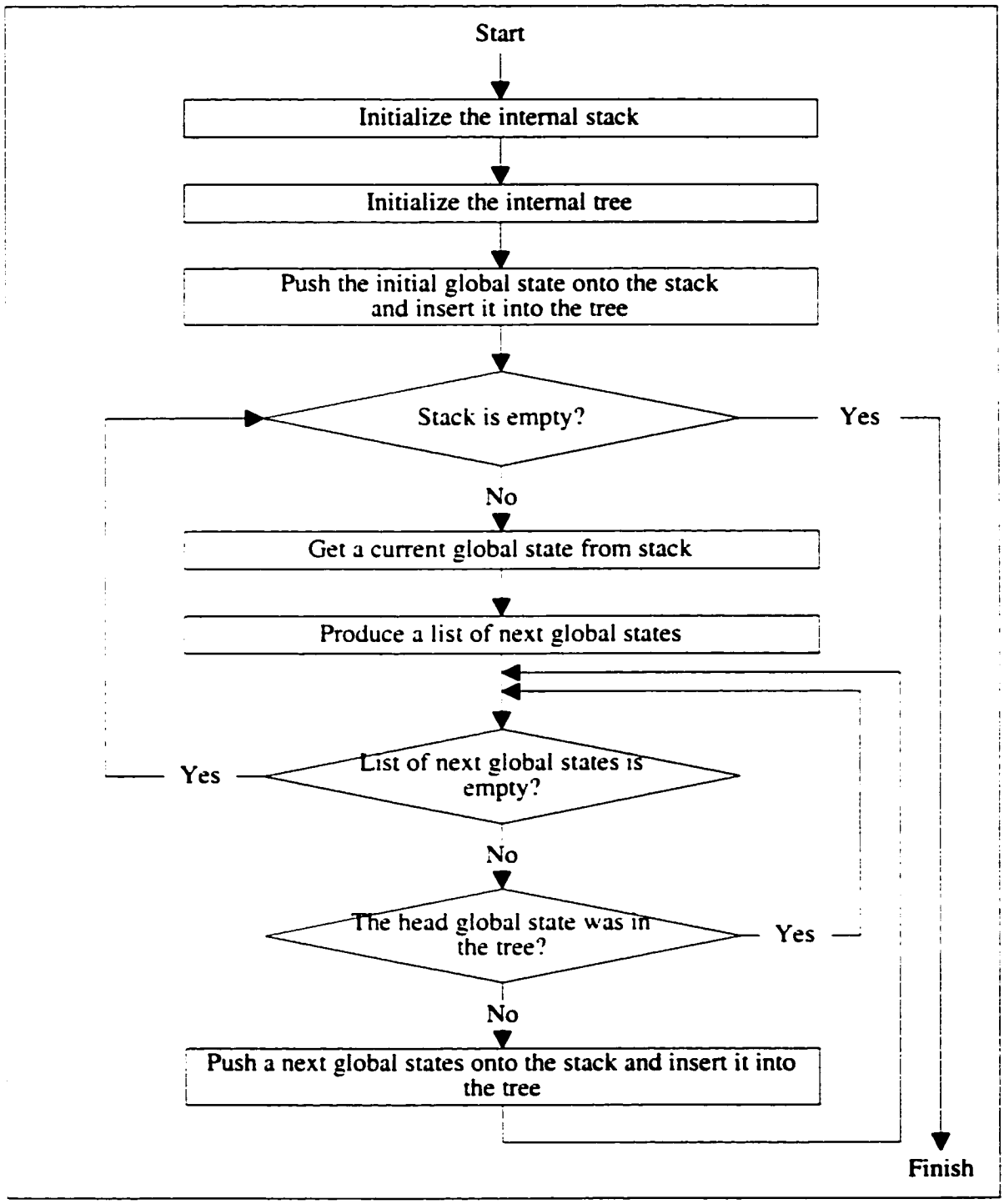

Figure 4.20 The generic procedure for reachability analysis

```
**********************************************************************
    <0,0>,<,>
    <0,1>,<,m2>
    <1,1>,<,>
    <2,1>,<m3,>
    <1,1>,<m1,m2>
    <1,2>,<,m2>
    <2,2>,<,>
    <2,2>,<m3,m2>
    <2,0>,<,m2>
    <2,1>,<,m2.m2>
    <1,1>,<,m2>
    <2,1>,<,>
    <2,1>,<m3,m2>
    <1,1>,<m3,>
    <2,1>,<m3.m3,>
    <1,0>,<,>
    <2,0>,<m3,>
    <1,2>,<m3,>
    <2,2>,<m3.m3,>
    <2,1>,<m1,>
    <2,1>,<m1.m3,m2>
    <1,1>,<m1.m3,>
    <1,0>,<m1,>
    <1,2>,<,>
    <2,2>,<m3,>
    <2,0>,<,>
    <2,1>,<,m2>
    <2,0>,<m1.m3,>
```

Figure 4.21 The exploring order of the sequential global states in protocol Π1

The internal stack structured is defaulted as a static array. The stack array does not keep the information about the reachability path and this is sufficient for CRA and a number of relief strategies such as SRA, FRA, and ERA. Since RELIEFv3.0, some other relief strategies such as partial-order verification based on ample sets with stack closing (POVAS) in [ScUr98b], leaping reachability analysis with stack closing (LRASC) in [ScUr96b] that need to use the reachability path information during global state exploration. In this case, a dynamic stack list structure is made active to replace the static stack array structure. Then, a global state is removed from the stack list only after it is fully explored, i.e., all of the successors of that global state in the reachability graph are already explored. The common utility function *CU_TurnStackClosingOn*() is available in RELIEF to make the stack list active.

In general, the implementation of a relief strategy is placed in the application layer that interacts with the core implementation of the protocol validation unit by the

common utility functions. That layout of the RELIEF tool is to ease the process of adding more relief strategies and to avoid software couplings. The core implementation of RELIEF is rather stable, designers only need to add the procedures at the application layer when a new relief strategy needs to be implemented.

4.1.3 The Automatic Protocol Synthesizer

The aim of protocol synthesis methods is to construct a correct protocol from an incomplete protocol description. Synthesis methods can be classified as service oriented and non-service oriented synthesis methods [PrSa91]. Service oriented methods construct functionally correct protocols while non-service oriented methods construct logically correct protocols. The protocols synthesized by the automatic protocol synthesizer (APS) are used for the empirical study on the performances of the relief strategies, therefore APS applies a non-service oriented method.

APS is required to construct n CFSMs such that the compositional behavior of these processes includes a significant number of sequentially reachable global states, and very limited number of logical errors for the purpose of comparing the relief strategies with CRA. APS uses a set of parameters such as *MinP*, *MaxP*, *MinS*(n), *MaxS*(n) where n=*MinP*, *MinP*+1, ..., *MaxP* and *MinP*, *MaxP* are the minimum, maximum number of processes in a protocol, respectively. For each n-process protocol, *MinS*(n) and *MaxS*(n) is the minimum and maximum number of states in a process. The values of these parameters are currently hard-coded as follows: *MinP*=2, *MaxP*=8, *MinS*(n)=2, and *MaxS*(n)=18−2*(n-2) for n=*MinP*, *MinP*+1, ..., *MaxP*.

APS requires a random number generator function *RS*(*low*, *high*) which randomly select an integer between low and high, where *low* and *high* are integers and *low*≤*high*. This function is implemented in C using random number generator function lrand48() in Unix. APS consists of two main procedures:

- The first procedure produces an incomplete specification of the protocol by randomly specifying transition function for sending transitions.

- The second procedure completes the protocol specification by specifying necessary receiving transitions using conventional reachability analysis.

Constructing protocols from an incomplete description by using conventional reachability analysis is also used in some protocol synthesis methods [Si82, ZaTaShNo88]. Such a synthesis method does not prevent deadlock and buffer overflows. In order to generate unspecified receptions and non-executable transitions, the specification of twenty-five per cent of necessary receiving transitions is ignored in the second procedure. Thus, it is guaranteed that the synthesized protocols can contain every type of logical errors. As expected, the number of global states of some of the synthesized protocols is either too small or too large. Such protocols are eliminated by setting lower and upper bounds on the number of global states. For n-process protocols, the lower bound is $500*2^n$ and the upper bound is 300000 global states.

4.1.4 The Empirical Study Tools

This component of the RELIEF package consists of various tools to support the empirical study on the relief strategies implemented in the protocol validation unit. Currently, the empirical study tools provide the following facilities:

- To pick up a multi-protocol input file synthesized by APS and repeatedly send it to the protocol validation unit to get the validation output files either by CRA or by various relief strategies.
- To analyze a multi-protocol input file, which is synthesized by APS, to report the properties and the CRA results of the protocols in the set.
- To extract the multi-cyclic protocols from a set of APS protocols.
- To compare the performance of a relief strategy with CRA and show the reduction percentage in generated global states, global transitions, used memory space, and execution time for that relief strategy.

4.2 The RELIEF Empirical Study

4.2.1 The Set of Four-Hundred APS Protocols

A set of four-hundred protocols synthesized by APS is used for the empirical study on every relief strategies implemented in the protocol validation unit. For every APS protocol in this set, the channel bounds are set to three messages. One of the empirical study tools is used to analyze the properties of the APS protocols in this set and the statistical results are reported in Table 4.1. APS synthesizes roughly equal number of protocols for each value of n. For each n-process protocol, the structural properties of processes such as the number of process states, the number of sending transitions and the number receiving transitions disperse pretty well. This can be seen from Table 4.1 where standard deviations for those properties are relatively high w.r.t. their averages.

Number of processes in a protocol							
Average	4.80						
StD	1.95						
***n*-process protocols**							
***n*=**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Number of protocols							
	66	57	63	58	56	60	40
Number of processes communicating with a process							
Average	1.00	1.47	1.44	1.53	1.77	1.77	1.40
StD	0.00	0.50	0.56	0.68	0.74	0.76	0.54
Number of states in a process							
Average	11.64	9.06	8.10	7.07	6.12	4.95	3.99
StD	4.21	4.10	3.47	3.13	2.45	1.98	1.43
Number of transitions in a process state							
Average	2.87	2.12	1.73	1.66	1.45	1.41	1.27
StD	1.43	1.47	1.47	1.41	1.19	1.19	0.99
Number of sending transitions in a process state							

Average	0.68	0.66	0.67	0.66	0.65	0.65	0.63
StD	0.71	0.70	0.71	0.70	0.66	0.65	0.58
Number of receiving transitions in a process state							
Average	2.18	1.46	1.06	1.01	0.81	0.76	0.64
StD	1.27	1.31	1.30	1.24	1.03	1.03	0.84
Percentage of sending states in a process (%)							
Average	3.60	16.95	23.29	26.02	28.12	30.61	33.13
StD	7.57	23.05	23.55	25.36	25.06	25.51	26.41
Percentage of receiving states in a process (%)							
Average	42.60	32.66	27.15	24.40	24.93	22.40	22.16
StD	10.75	19.41	20.36	20.38	20.60	21.15	20.92
Percentage of mixed-action states in a process (%)							
Average	52.30	38.56	31.69	28.98	27.64	25.13	23.85
StD	12.12	23.63	25.73	25.96	26.17	26.14	26.53
Percentage of null states in a process (%)							
Average	1.50	11.82	17.87	20.60	19.30	21.87	20.86
StD	4.40	18.12	21.12	20.52	20.92	22.20	21.74

Table 4.1 The properties of the set of four-hundred APS protocols

The CRA results of the APS protocols are summarized in Table 4.2. The standard deviations show that the numbers of sequentially reachable global states of the APS protocols disperse very well in each class of n-process protocols.

***n*-process protocols**							
***n*=**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Number of sequential global states (*100)							
Average	31.71	34.02	49.09	60.87	78.37	107.63	149.99
StD	39.56	37.60	41.11	40.00	37.78	32.59	13.09
Number of sequential global transitions (*100)							
Average	69.21	100.77	171.29	210.95	319.40	467.00	764.27
StD	86.17	125.80	151.56	154.59	176.05	161.94	87.49

Memory space (Kbytes)							
Average	1667.28	1840.90	2671.01	3312.63	4264.46	5908.87	8279.19
StD	2084.58	2067.01	2258.00	2210.80	2089.85	1793.92	771.64
Execution time (Seconds)							
Average	1.22	1.67	2.93	3.88	6.06	9.81	16.41
StD	1.74	2.21	2.84	3.03	3.58	3.63	1.91
Percentage of blocking states (%)							
Average	3.46	1.90	1.04	1.12	0.34	0.20	0.04
StD	1.98	1.54	1.19	1.23	0.59	0.28	0.06
Percentage of deadlock states (%)							
Average	0.23	0.15	0.08	0.04	0.02	0.01	0.00
StD	0.32	0.17	0.10	0.05	0.02	0.01	0.00
Percentage of unspecified reception causing pairs (%)							
Average	21.00	21.01	19.74	22.17	17.66	17.70	15.13
StD	6.49	5.51	7.70	7.17	6.71	6.86	5.12
Percentage of buffer overflow causing pairs (%)							
Average	24.27	22.47	20.34	16.97	15.56	14.33	15.41
StD	5.03	6.00	6.07	5.61	5.08	5.42	5.87
Percentage of non-executable transitions (%)							
Average	0.45	5.09	10.48	11.48	13.70	14.59	13.46
StD	1.43	5.30	7.60	8.04	8.34	8.36	6.10
Average number of concurrent processes at a global state							
Average	0.56	1.17	1.51	1.57	1.94	2.03	2.62
StD	0.03	0.16	0.40	0.42	0.49	0.49	0.34
Average number of executable transitions at a global state							
Average	2.17	2.74	3.32	3.35	3.98	4.32	4.96
StD	0.13	0.46	0.54	0.55	0.48	0.48	0.40

Table 4.2 The CRA results of the set of four-hundred APS protocols

Table 4.2 includes results on the amount of execution time and logical errors detected by CRA. Also reported is the average number of processes having executable transitions at a global state in a protocol. This average can be viewed as the *concurrency level* in a protocol and used as a scale for the study on the performance of a relief strategy beside the number of processes.

Yet, another empirical study tool can be used to extract the multi-cyclic protocols from the set of four-hundred APS protocols. This set of multi-cyclic protocols is used for the empirical study on the relief strategies such as FRA. The properties of the multi-cyclic protocols in this set and the statistical results are reported in Table 4.3.

Number of processes in a protocol							
Average	2.31						
StD	0.59						
***n*-process protocols**							
***n*=**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Number of protocols							
	66	18	3	1			
Number of processes communicating with a process							
Average	1.00	1.50	2.00	2.00			
StD	0.00	0.50	0.00	0.00			
Number of states in a process							
Average	11.64	8.91	8.33	8.00			
StD	4.21	4.35	3.14	2.00			
Number of transitions in a process state							
Average	2.87	2.35	1.75	1.65			
StD	1.43	1.47	1.10	1.26			
Number of sending transitions in a process state							
Average	0.68	0.66	0.71	0.70			
StD	0.71	0.72	0.77	0.78			

Number of receiving transitions in a process state				
Average	2.18	1.69	1.04	0.95
StD	1.27	1.32	0.94	1.07
Percentage of sending states in a process (%)				
Average	3.60	11.40	20.88	28.27
StD	7.57	18.44	17.08	24.25
Percentage of receiving states in a process (%)				
Average	42.60	38.45	33.59	27.28
StD	10.75	14.79	17.32	17.61
Percentage of mixed-action states in a process (%)				
Average	52.30	42.34	34.77	24.60
StD	12.12	19.61	20.41	21.28
Percentage of null states in a process (%)				
Average	1.50	7.81	10.76	19.85
StD	4.40	13.47	20.19	13.66

Table 4.3 The properties of the set of multi-cyclic protocols

The CRA results of the multi-cyclic protocols are summarized in Table 4.4.

***n*-process protocols**							
***n*=**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Number of sequential global states (*100)							
Average	31.71	50.21	21.68	162.57			
StD	39.56	43.97	6.04	0.00			
Number of sequential global transitions (*100)							
Average	69.21	146.86	74.75	565.48			
StD	86.17	133.26	31.57	0.00			
Memory space (Kbytes)							
Average	1667.28	2736.38	1163.58	9019.50			
StD	2084.58	2403.27	339.11	0.00			
Execution time (Seconds)							

Average	1.22	2.55	1.05	10.62
StD	1.74	2.49	0.41	0.00
Percentage of blocking states (%)				
Average	3.46	2.13	0.90	0.54
StD	1.98	1.63	0.61	0.00
Percentage of deadlock states (%)				
Average	0.23	0.16	0.14	0.03
StD	0.32	0.20	0.12	0.00
Percentage of unspecified reception causing pairs (%)				
Average	21.00	22.50	19.01	15.15
StD	6.49	4.49	3.95	0.00
Percentage of buffer overflow causing pairs (%)				
Average	24.27	20.03	16.15	18.18
StD	5.03	6.42	2.64	0.00
Percentage of non-executable transitions (%)				
Average	0.45	3.10	11.03	6.06
StD	1.43	5.30	7.60	0.00
Average number of concurrent processes at a global state				
Average	0.56	1.20	1.36	1.42
StD	0.03	0.15	0.36	0.00
Average number of executable transitions at a global state				
Average	2.17	2.77	3.33	3.48
StD	0.13	0.47	0.52	0.00

Table 4.4 The CRA results of the set of multi-cyclic protocols

4.2.2 The RELIEF Empirical Results

Typically, a report from the empirical study tools for the performance of a relief strategy compared with CRA is based on two scales: the number of processes in a protocol and the levels of concurrency. For each scale, the empirical study tools report the average reduction in stored global states, the average reduction in generated global transitions, the

average reduction in memory space used to store the global states, and the average reduction in execution time.

Table 4.5 is for the performance of the leaping reachability analysis (LRAd) compared with CRA using the set of four-hundred APS protocols (See Table 4.1 and Table 4.2). From the corresponding empirical study, all deadlocks in the four-hundred protocols, which are revealed by CRA are also revealed by the LRAd strategy.

- In terms of the number of processes in a protocol: the reduction in stored global states is increasing from 55.94 per cent in the average for two-process protocols up to 94.10 per cent in the average for eight-process protocols; the reduction in generated global transitions is increasing from 65.49 per cent in the average for two-process protocols up to 97.36 per cent in the average for eight-process protocols.

- In terms of the concurrency levels: the reduction in stored global states is increasing from 53.77 per cent in the average when the concurrency levels are in the [0,1] interval up to 97.94 per cent in the average when the concurrency levels are in the (3,4] interval; the reduction in generated global transitions is increasing from 64.63 per cent in the average when the concurrency levels are in the [0,1] interval up to 99.19 per cent in the average when the concurrency levels are in the (3,4] interval.

***n*-process protocols**							
***n*=**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Reduction in global states (%)							
Average	55.94	64.65	72.36	75.68	83.54	84.75	94.10
Reduction in global transitions (%)							
Average	65.49	74.76	81.77	85.02	90.76	91.61	97.36
Reduction in memory space (%)							
Average	55.70	64.70	72.41	75.81	83.68	84.88	94.15
Reduction in execution time (%)							

Average	59.04	65.76	73.73	77.12	84.52	85.11	94.63

Levels of concurrency					
	[0, 1]	**(1, 2]**	**(2, 3]**	**(3, 4]**	**(4, 5]**
Reduction in global states (%)					
Average	53.77	74.22	92.01	97.94	
Reduction in global transitions (%)					
Average	64.63	83.50	96.25	99.19	
Reduction in memory space (%)					
Average	53.63	74.31	92.08	97.92	
Reduction in execution time (%)					
Average	55.91	75.48	92.89	98.28	

Table 4.5 The performance of LRAd compared with CRA by RELIEF

Table 4.6 is for the performance of the fair reachabilty analysis (FRA) in [ScUr95b] compared with CRA. Of course, the set of multi-cyclic protocols is used in this case (See Table 4.3 and Table 4.4). From the corresponding empirical study, all deadlocks in the multi-cyclic protocols, which are revealed by CRA are also revealed by the FRA strategy.

- In terms of the number of processes in a protocol: the reduction in stored global states is increasing from 77.02 per cent in the average for two-process protocols up to 97.46 per cent in the average for five-process protocols; the reduction in generated global transitions is increasing from 80.47 per cent in the average for two-process protocols up to 98.07 per cent in the average for five-process protocols.

- In terms of the concurrency levels: the reduction in stored global states is increasing from 77.60 per cent in the average when the concurrency levels are in the [0,1] interval up to 93.27 per cent in the average when the concurrency levels are in the (1,2] interval; the reduction in generated global

transitions is increasing from 81.04 per cent in the average when the concurrency levels are in the [0,1] interval up to 95.48 per cent in the average when the concurrency levels are in the (1,2] interval.

***n*-process protocols**							
***n*=**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Reduction in global states (%)							
Average	77.02	92.41	94.23	97.46			
Reduction in global transitions (%)							
Average	80.47	94.89	96.29	98.07			
Reduction in memory space (%)							
Average	76.69	92.50	94.44	97.60			
Reduction in execution time (%)							
Average	-24.49	60.39	70.71	88.89			

Levels of concurrency					
	[0, 1]	**(1, 2]**	**(2, 3]**	**(3, 4]**	**(4, 5]**
Reduction in global states (%)					
Average	77.60	93.27			
Reduction in global transitions (%)					
Average	81.04	95.48			
Reduction in memory space (%)					
Average	77.30	93.35			
Reduction in execution time (%)					
Average	-21.75	66.98			

Table 4.6 The performance of FRA compared with CRA by RELIEF

Some information about RELIEF can be also found at URL http://www.site.uottawa.ca/~ural/relief/ at the present time. The RELIEF package has been being extended from one version to another version to support continuously the

research on reachability analysis. The tool helps standardize a study on a relief strategy from initial ideas to final reports. The core implementation of RELIEF is rather stable, designers only need to add the procedures at the application layer when a new experiment is needed. Empirical results can be obtained shortly to verify a new experiment. In the following chapters, the experiments on various relief strategies that are all called even reachability analysis (ERA) will be carried out with the support from the RELIEF package.

Chapter 5 The ERAp Strategy

During the fourth international conference on computer communications and networks (ICCCN'95), which was held in Las Vegas in September 1995, Peng introduced another relief strategy for reachability analysis, called *even reachability analysis* [Pe95]. In this thesis, the ERAp abbreviation is used to distinguish the Peng strategy with the other strategy of even reachability analysis that is to be discussed in the next chapter.

The ERAp strategy is characterized in the above proceedings as follows:

- Applicable to the general class of CFSMs.
- Forced two process transitions during global state exploration.
- All reachable global states by ERAp having an even total length of all the channels.
- An extension of the classical fair reachability analysis [GoHa85].
- Deadlock freedom is verifiable.
- More than one half of the total number of the sequentially reachable global states being saved.

This chapter reviews the related paper ([Pe95]), identifies its shortcomings, and demonstrates that the ERAp strategy does not provide a full coverage of deadlock states. The sections in this chapter are arranged as follows: Section 5.1 defines the ERAp preliminaries, Section 5.2 characterizes the ERAp strategy, Section 5.3 examines the mathematical support in [Pe95] for the ability of ERAp in verifying deadlock freedom of a protocol, and Section 5.4 shows the failures of ERAp using a counter-example.

5.1 The ERAp Preliminaries

5.1.1 Even Reachable Global State Space and Stable Global State Space

Definition 5.1 An *even reachable global state* is a sequentially reachable global state in the CFSM model where the total length of all the channels is an even number, which is called *even total channel length property*, i.e., $\forall G_k \in \Gamma$, $\forall i, j \in I \wedge i \neq j$ (G_k is an even reachable global state $\Leftrightarrow \sum_{i,j} |c_{ij}|$ is an even number). The even reachable global state space is denoted by Γ_{EVEN}. □

Definition 5.2 A *stable global state* is a sequentially reachable global state in the CFSM model where all channels are empty, which is called *stable channel length property*, i.e.,

- $\forall G_k \in \Gamma$, $\forall i, j \in I \wedge i \neq j$ (G_k is a stable global state $\Leftrightarrow c_{ij}=\varepsilon$) or
- $\forall G_k \in \Gamma$, $\forall i, j \in I \wedge i \neq j$ (G_k is a stable global state $\Leftrightarrow \sum_{i,j} |c_{ij}|=0$).

The stable global state space is denoted by Γ_{STABLE}. □

According to these definitions, a stable global state is an even reachable global state but the reverse is not always true. Moreover, a deadlock state as defined in Definition 2.10 is a stable global state but again, the reverse is not always true. Those relations can be illustrated as in Figure 5.1 and summarized as follows: $\forall G_k \in \Gamma$ (G_k is a deadlock state $\Rightarrow G_k \in \Gamma_{STABLE} \Rightarrow G_k \in \Gamma_{EVEN}$).

Suppose that some relief strategy armed with some suitably defined rules of global transitions could cover the even reachable global state space, or even better cover only the stable global state space, then it would successfully verify deadlock freedom of a protocol without exploring entirely the sequentially reachable global states space Γ. In light of this, the ERAp strategy, starting from the initial global state, forces a pair of two process transitions to be executed at a time. Then, one would expect that the strategy preserves the even channel length property in the generated global states and thus, it

explores only part of the even reachable global state space to verify deadlock freedom of a protocol. The reachable global state space by ERAp is denoted by Γ_{ERAp}, which is a subset of Γ_{EVEN}. A theorem provided in [Pe95] attempts to claim that every stable global state is reachable by ERAp using the *selected ERAp transition pairs*, i.e., $\Gamma_{STABLE} \subset \Gamma_{ERAp}$. Before judging the correctness of the theorem, we will give the definitions of the ERAp transition pairs (i.e., Definition 5.3 to Definition 5.6) in our terminology.

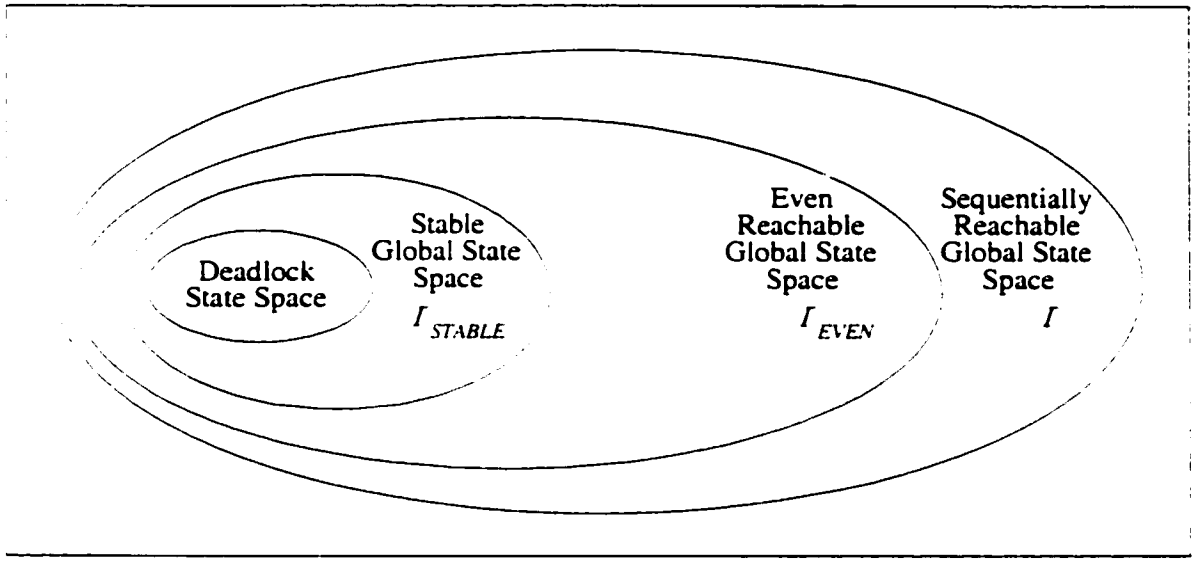

Figure 5.1 Illustration of the relations in Definition 5.1 and Definition 5.2

5.1.2 The ERAp Transition Pairs

A *type-1 ERAp transition pair* defined in [Pe95] is a pair of two transitions from any two different processes P_i and P_j such that, the process transition from P_i is a transmission of a message from P_i to P_j and the process transition from P_j is a reception of a message from P_i by P_j.

Although [Pe95] does not explicitly mention about potentially executable transitions (Definition 2.11) in the construction of type-1 ERAp transitions, the two examples in that paper however suggest that type-1 ERAp transition pairs should include the potentially executable receiving transitions as well. Therefore, we will consider potentially executable transitions in our redefinition that we give in our terminology.

Recall that potentially executable receiving transitions are called enabled transitions in [LiMi94a].

Definition 5.3 A *type-1 ERAp transition pair* at a global state G_k, which is denoted by τ^1, is a pair of two process transitions constructed by one of the following conditions:

1. $(\exists i, j \in I \wedge i \neq j \wedge \sigma_{ij}, \rho_{ij} \in exec(G_k)) \Rightarrow \tau^1 = (\sigma_{ij}, \rho_{ij})$.
2. $(\exists i, j \in I \wedge i \neq j \wedge \sigma_{ij} \in exec(G_k) \wedge \rho_{ij} \in potent(G_k)) \Rightarrow \tau^1 = (\sigma_{ij}, \rho_{ij})$. □

The definitions of σ_{ij}, ρ_{ij} are already provided in Definition 2.2. The order of the transitions in a pair is not a matter in Condition 1, but it is a matter in Condition 2. In addition, since τ^1 always consists of two opposite actions involving the same channel, then the execution of τ^1 does not change the length of the messages in the involved channel.

A *type-2 ERAp transition pair* defined in [Pe95] is an unordered pair of two transitions from any two different processes P_i and P_j such that, the process transition from P_i is a message transmission and the process transition from P_j is message reception.

To avoid overlapping between type-1 and type-2 ERAp transition pairs, the above definition should specify that the two process transitions must involve two different channels. Again, although the definition in [Pe95] only mentions the case of one transmission and one reception, the case of two transmissions from two different processes is used in the proof of Lemma 3.1 in [Pe95].

Definition 5.4 A *type-2 ERAp transition pair* at a global state G_k, which is denoted by τ^2, is a pair of two process transitions constructed by one of the following conditions:

1. $(\exists i, j, p, q \in I \wedge i \neq p \wedge j \neq q \wedge i \neq j \wedge \sigma_{ip}, \rho_{qj} \in exec(G_k)) \Rightarrow \tau^2 = (\sigma_{ip}, \rho_{qj})$.
2. $(\exists i, j, p, q \in I \wedge i \neq p \wedge j \neq q \wedge i \neq j \wedge \sigma_{ip}, \sigma_{jq} \in exec(G_k)) \Rightarrow \tau^2 = (\sigma_{ip}, \sigma_{jq})$. □

Notice that [Pe95] leaves out the pairs of two receptions from two different processes without providing any reasons. Since they are not defined or used anywhere in the paper, the redefinition of type-2 ERAp transition pairs that we give in our terminology cannot add them to the original definition.

A *type-3 ERAp transition pair* defined in [Pe95] is a pair of two consecutive transitions from the same process P_i.

Because [Pe95] does not specify the types of actions when mentioning two consecutive transitions in the above definition, we feel that all possible combinations should be exhausted in our redefinition in order to give the benefit of the doubt to the Peng strategy.

Definition 5.5 A *type-3 ERAp transition pair* at a global state G_k, which is denoted by τ^3, is an ordered pair of two process transitions constructed by one of the following conditions:

1. $(\exists i, j \in I \wedge i \neq j \wedge \sigma_{ij} \in exec(G_k) \wedge \exists G_l \in \Gamma\ G_k \text{—}\sigma_{ij}\rightarrow G_l \wedge \exists p \in I \wedge i \neq p \wedge \sigma_{ip} \in exec(G_l)) \Rightarrow \tau^3 = (\sigma_{ij}, \sigma_{ip})$.

2. $(\exists i, j \in I \wedge i \neq j \wedge \sigma_{ij} \in exec(G_k) \wedge \exists G_l \in \Gamma\ G_k \text{—}\sigma_{ij}\rightarrow G_l \wedge \exists p \in I \wedge i \neq p \wedge \rho_{pi} \in exec(G_l)) \Rightarrow \tau^3 = (\sigma_{ij}, \rho_{pi})$.

3. $(\exists i, j \in I \wedge i \neq j \wedge \rho_{ji} \in exec(G_k) \wedge \exists G_l \in \Gamma\ G_k \text{—}\rho_{ji}\rightarrow G_l \wedge \exists p \in I \wedge i \neq p \wedge \sigma_{ip} \in exec(G_l)) \Rightarrow \tau^3 = (\rho_{ji}, \sigma_{ip})$.

4. $(\exists i, j \in I \wedge i \neq j \wedge \rho_{ji} \in exec(G_k) \wedge \exists G_l \in \Gamma\ G_k \text{—}\rho_{ji}\rightarrow G_l \wedge \exists p \in I \wedge i \neq p \wedge \rho_{pi} \in exec(G_l)) \Rightarrow \tau^3 = (\rho_{ji}, \rho_{pi})$. □

Notice that G_l is a sequential immediate successor of G_k as defined in Definition 2.7.

The set of all type-1, type-2, and type-3 ERAp transition pairs at a global state G_k is denoted by $exec_{ERAp}(G_k)$. Notice that the execution of an ERAp transition pair preserves the even total channel length property.

[Pe95] highlights the use of the selected ERAp transition pairs to generate global states as follows: "It is required that the three types of ERA transition pairs are considered in order. In other words, if a global state G_l is reachable from a global state G_k by a type-1 transition pair, and another global state G_m is reachable from the same G_k by a type-2 transition pair, then G_m is not considered reachable by ERAp from G_k. Similar restriction applies between type-2 and type-3 transition pairs, and between type-1 and type-3 transition pairs." Those ideas have been carefully captured in the construction of $select_{ERAp}(G_k)$, the set of all selected ERAp transition pairs at a global state G_k, which is provided in Definition 5.6.

Definition 5.6 $\tau \in exec_{ERAp}(G_k)$ is a *selected ERAp transition pair* at a global state G_k if one of the following conditions is satisfied:

1. τ is a type-1 ERAp transition pair, or
2. τ is a type-2 ERAp transition pair if there are no type-1 ERAp transition pairs in $exec_{ERAp}(G_k)$, or
3. τ is a type-3 ERAp transition if there are neither type-1 nor type-2 ERAp transition pairs in $exec_{ERAp}(G_k)$. □

The set of all selected ERAp transition pairs at a global state G_k is denoted by $select_{ERAp}(G_k)$. It is not difficult to see that all selected ERA transition pairs at a global state must be of the same type.

5.1.3 The ERAp Relations

In the ERAp reachability analysis of a protocol, a global state, say G_k, generates another global state, say G_l, if there exists a selected ERAp transition pair τ at G_k, and the two process transitions in that pair are executed to reach G_l.

Definition 5.7 A global state G_l is an *ERAp immediate successor* of a global state G_k, denoted by $G_k \rightarrow_{ERAp} G_l$ or $G_k \text{---}\tau\rightarrow_{ERAp} G_l$, iff $\exists \tau \in select_{ERAp}(G_k)$ such that $G_k \text{---}\tau\rightarrow^* G_L$. □

Notice that, in Definition 5.7, the ERAp immediate successor defines a binary relation between the reachable global states, which is denoted by $\rightarrow_{ERAp}$.

Definition 5.8 Let $\rightarrow_{ERAp}^*$ be the reflexive and transitive closure of $\rightarrow_{ERAp}$.

1. G_m is a reachable global state by ERAp from G_k (or an ERAp successor of G_k) iff $G_k \rightarrow_{ERAp}^* G_m$.

2. G_m is a reachable global state by ERAp iff $G_0 \rightarrow_{ERAp}^* G_m$.

$G_k \text{---}\tau_1\rightarrow_{ERAp} G_{k(1)} \wedge G_{k(1)} \text{---}\tau_2\rightarrow_{ERAp} G_{k(2)} \wedge \ldots \wedge G_{k(j-1)} \text{---}\tau_j\rightarrow_{ERAp} G_m$ is also denoted by $G_k \text{---}\tau_1 \tau_2 \ldots \tau_j\rightarrow_{ERAp}^* G_m$. □

Reachable global state space Γ_{ERAp} of a protocol by the ERAp strategy can be represented by a directed graph, called *ERAp reachability graph* and also denoted by Γ_{ERAp}, in which the nodes represent global states and the arcs stand for the ERAp immediate successor relation between global states, which will be also called global transitions in this thesis.

5.2 The Characterization of ERAp

First, [Pe95] does not make any restricting assumption for ERAp either on the topology of the protocol, the number of processes, the structure of the processes, or on the simplex channels at all. On the other side, ERAp attempts to restrict the exploration to a subset of the global state space of the protocol, where the absence of deadlock states needs to be verified. The subset of Γ which ERAp aims to is the stable global state space Γ_{STABLE} as shown in Figure 5.1. However, by preserving only the even total channel length property while producing the global states, the ERAp transitions may also generate some global states that do not belong to the stable global state space but the even reachable global state space Γ_{EVEN}.

[Pe95] claims that ERAp guarantees to verify deadlock freedom of any protocol. It would be obvious if the selected ERAp transition pairs defined in Definition 5.6 could cover the stable global state space. The attempted reachable global state space by ERAp is shown in Figure 5.2 using dashed lines.

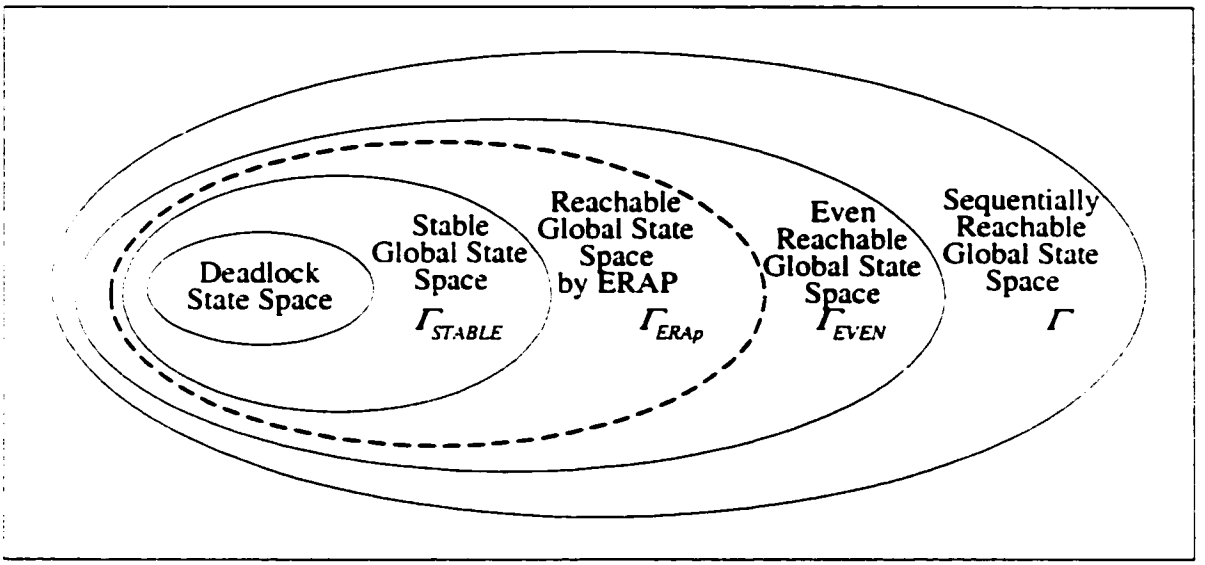

Figure 5.2 The attempted reachable global state space by ERAp

[Pe95] claims that ERAp is an extension of the fair reachability analysis proposed by Gouda and Han [GoHa85], and that a fair reachable global state implies a reachable global state by ERAp. Recall that the fair reachability analysis by Gouda and Han, which is summarized in Section 3.2.1.3, can only apply to the class of two-process protocols while ERAp may apply to the general class of n-process protocols. Although ERAp does not preserve the equal progress property of the fair reachability analysis, but for protocols with n=2 the even total channel length property must lead to the same reduced reachability graph as the equal progress property. Otherwise, ERAp has fewer things to do with fair reachability analysis.

5.3 ERAp and Deadlock Detection

The mathematical support for the ability of ERAp in verifying deadlock-freedom of a protocol provided in [Pe95] is summarized as follows:

1. Lemma 3.1: "If G_k is a reachable global state by ERAp, then G_k is a sequentially reachable global state."
2. Theorem 3.1: "If G_k is a stable global state, then G_k is a reachable global state by ERAp."
3. Corollary 3.1: "A protocol is free from deadlocks iff all stable global states that are reachable by ERAp are free from deadlocks."

The numbering is taken from [Pe95]. The rest of this section examines each of those statements.

Point 5.1 A criticism on the proof for Lemma 3.1 in [Pe95]: "If G_k is a reachable global state by ERAp, then G_k is a sequentially reachable global state."

In [Pe95], a mathematical induction on the number of transition steps is provided as a proof for this lemma. Interestingly, from the mathematical induction, two sending transitions from two different processes are selected to construct the first ERAp transition pair in the proof of the lemma without previously defining this case. These ERAp transition pairs are accordingly added to the redefinition of type-2 ERAp transition pairs in Definition 5.4. Below, we give a clearer proof for this lemma:

Suppose that G_k is a reachable global state by ERAp,

$\Rightarrow G_0 \rightarrow_{ERAp}{}^* G_k$ (by Definition 5.8)

$\Rightarrow \exists(\tau_1 \tau_2 \ldots \tau_j)\ G_0 \text{—} \tau_1 \tau_2 \ldots \tau_j \rightarrow_{ERAp}{}^* G_k$

$\Rightarrow \exists(t_1^1, t_1^2, t_2^1, t_2^2, \ldots, t_j^1, t_j^2)\ \tau_1=(t_1^1, t_1^2),\ \tau_2=(t_2^1, t_2^2),\ \ldots,\ \tau_j=(t_j^1, t_j^2)$

(by Definition 5.6)

$\Rightarrow \exists(t_1^1 t_1^2 t_2^1 t_2^2 \ldots t_j^1 t_j^2)\ G_0 \text{—} t_1^1 t_1^2 t_2^1 t_2^2 \ldots t_j^1 t_j^2 \rightarrow {}^* G_k$

$\Rightarrow G_0 \rightarrow {}^* G_k$

then, G_k is a sequentially reachable global state by Definition 2.8. □

Point 5.2 A criticism on the proof for Theorem 3.1 in [Pe95]: "If G_k is a stable global state, then G_k is a reachable global state by ERAp."

First of all, the proof of this theorem is claimed to be a mathematical induction on the number of transition steps $2(k)$, where $k \geq 0$:

- Basis step: The stable global state at $2(k=0)$-transition step is obviously the initial global state G_0. By Definition 5.8, $G_0 \rightarrow_{ERAp}{}^* G_0$, i.e., G_0 is a reachable global state by ERAp.
- Assumption step: Assume that a stable global state at $2(k \geq 0)$-transition steps is a reachable global state by ERAp.
- Induction step: Derive from the previous assumption to prove that a stable global state at $2(k+1)$-transition steps is a reachable global state by ERAp. The reasoning for this step in [Pe95] is ill-formed because it does not derive anything from the assumption step, but looks more like a stand-alone mathematical contradiction.

Now, examine the proof as a stand-alone mathematical contradiction. Suppose that G_{k+1} is a stable global state that is sequentially reachable by executing $2(k+1)$-transition steps, i.e.,

$$\Rightarrow G_0 \rightarrow^* G_{k+1}$$

$$\Rightarrow \exists(\, t_1{}^1 t_1{}^2 t_2{}^1 t_2{}^2 \ldots t_{k+1}{}^1 t_{k+1}{}^2)\; G_0 \text{—} t_1{}^1 t_1{}^2 t_2{}^1 t_2{}^2 \ldots t_{k+1}{}^1 t_{k+1}{}^2 \rightarrow^* G_{k+1}$$

[Pe95] takes as a contradictory claim that it is impossible to move forward from the initial global state G_0 by using the selected ERAp transition pairs derived from the original sequence $t_1{}^1 t_1{}^2 t_2{}^1 t_2{}^2 \ldots t_{k+1}{}^1 t_{k+1}{}^2$. Then, it attempts to prove that, in all possible situations, there always exists a selected ERAp transition pair derived from the original sequence to move forward until all process transitions in the original sequence are exhausted. Hence, a contradiction arises to conclude the proof. Unfortunately, the contradiction does not always arise as seen in the counter-example that is provided in

Section 5.4. In addition, notice that an algorithm for deriving the selected ERAp transition pairs from the original sequence of process transitions is not provided in [Pe95]. □

Point 5.3 A criticism on Corollary 3.1 in [Pe95]: "A protocol is free from deadlocks iff all stable global states that are reachable by ERAp are free from deadlocks."

This is a direct result from Theorem 3.1. Since the proof of Theorem 3.1 is not convincing, then this corollary is hard to be accepted. □

5.4 The ERAp Failure Cases

Protocol Π1 specified in Figure 2.2 of Example 2-1 is used again as a counter-example to the ERAp strategy. Figure 5.3 shows the ERAp reachability graph of this two-process protocol. The sequentially reachable global states, which are not reachable by the Peng reachability analysis, are left blank in place for comparison purposes. Complete conventional reachability graph and fair reachability graph of protocol Π1 are already given in Figure 3.2 and Figure 3.3, respectively. Recall that, in a two-process protocol, fair reachability graph is the same no matter it is from the Rubin-West, the Liu-Miller, or the van der Schoot-Ural strategy. The ERAp report from the RELIEF tool for protocol Π1 is also captured in Figure 5.4. A number of points can be derived from this counter-example:

Point 5.4 If the ERAp strategy is a generalization of fair reachability analysis, then the reachable global state space by ERAp must include the fair reachable global state space for any 2-process protocol. Unfortunately, it is not the case when comparing the two reachability graphs in Figure 3.3 and Figure 5.3. More specifically, global state (<1, 1>,<m1, m2>) is a fair reachable global state but it is not reachable by ERAp. Therefore, the claim in [Pe95]: "In fact, even reachability is weaker than fair reachability: if a global state is fair reachable, it is also even reachable. The reverse, however, is not true." is in fact not true. □

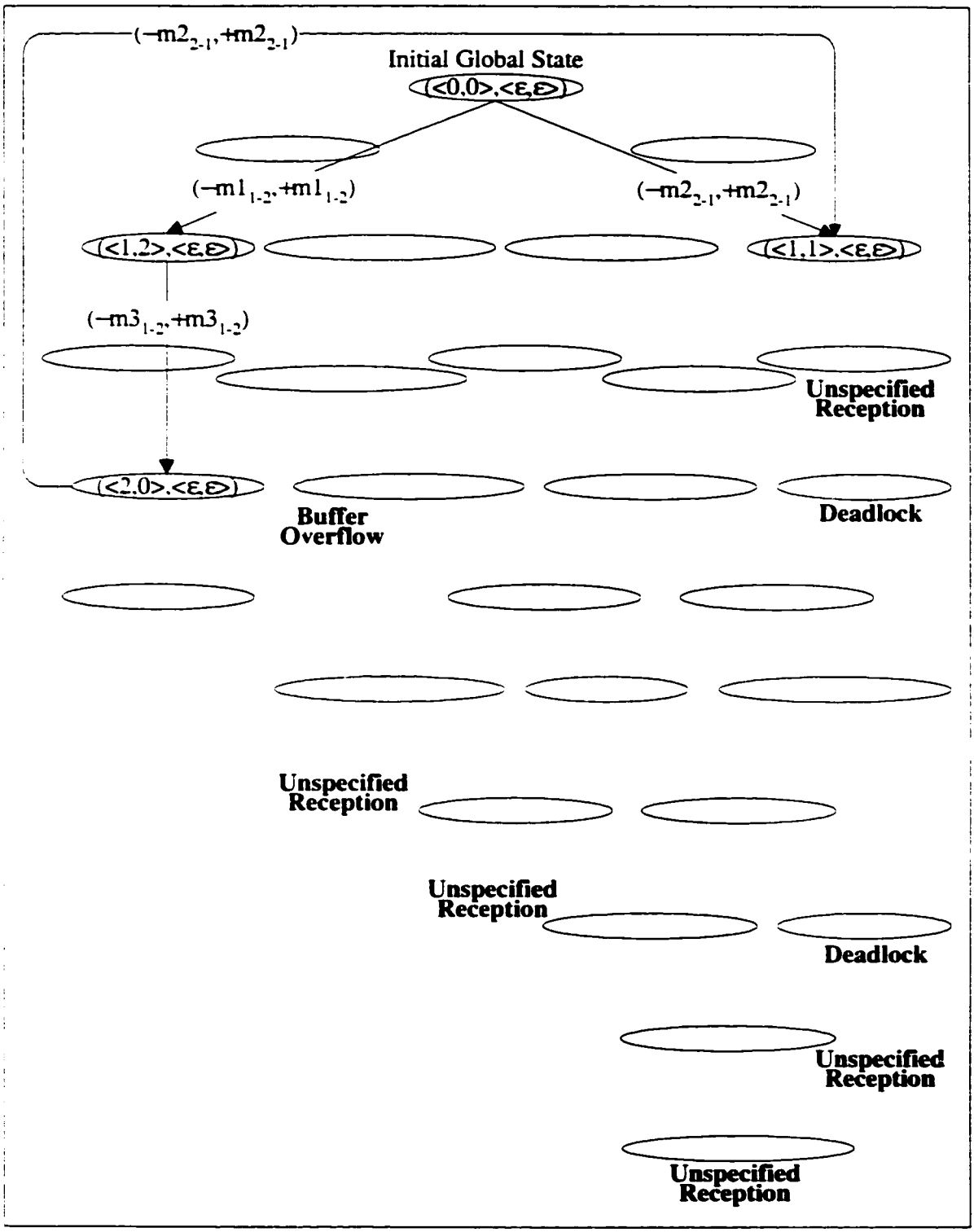

Figure 5.3 The ERAp reachability graph of two-process protocol Π1

```
**********************************************************************
                    EVEN REACHABILITY ANALYSIS (ERA)

      Full coverage.....................................(f)
      Peng coverage.....................................(p)


      File Utilities....................................(u)
      Back to Previous Menu.............................(b)
      Main Menu.........................................(m)
      Exit..............................................(x)

      Choice............................................(p

Analyzing Protocol No. 1, wait...done

Read report (y/n)? y

EVEN REACHABILITY ANALYSIS (ERAp)
Deadlock Detector reports:

PROTOCOL No.: 1
NUMBER OF PROCESSES: 2
Process No. 1: States Nos.: 0 1 2 3
Process No. 2: States Nos.: 0 1 2 3
NUMBER OF  CHANNELS: 2
Channel Bound: 2
Channel  Nos.: 1-2 2-1

  NUMBER OF PROTOCOL DESIGN ERRORS:
  **********************************************************************
      Blocking     Deadlock     UR Causing     BO Causing     Non-Executable
      States       States       (process,M)    (process,M)    Transitions
                                Pairs          Pairs
  **********************************************************************
      >= 0         0            >= 0           >= 0             ?

  Global States      : 4
  Global Transitions: 4
  Space Consumed     : 0.21 (KBytes)
  Time Consumed      : 0.00 (Seconds)
```

Figure 5.4 The ERAp results reported by RELIEF for two-process protocol Π1

There are two reasons for this. The first one is that the pairs of two receptions from two different processes are missing in the construction of type-2 ERAp transition pairs (See Definition 5.4). The second reason is that even if the pairs of two receptions from two different processes were added to type-2 ERAp transition pairs, the way to select the ERAp transition pairs (Definition 5.6) would still leave them out if there exists a type-1 ERAp transition pair at the current global state.

Point 5.5 Global state (<2, 2>,<ε, ε>) is a stable global state in protocol Π1, which is reachable from the initial global state by a sequence of process transitions ($-m1_{1\text{-}2}$, $-m2_{2\text{-}1}$, $+m1_{1\text{-}2}$, $+m2_{2\text{-}1}$). This stable global state is also a deadlock state. However, a sequence of selected ERAp transition pairs cannot be derived from these process transitions, contrary to the claim stated in Theorem 3.1 in [Pe95]:

- If ($-m1_{1\text{-}2}$, $+m1_{1\text{-}2}$) is selected as the first pair of transitions, then it is impossible to select the second pair of transitions from the remaining process transitions in the sequence to move forward on the ERAp reachability graph.
- If ($-m2_{2\text{-}1}$, $+m2_{2\text{-}1}$) is selected as the first pair of transitions, then it is impossible to select the second pair of transitions from the remaining process transitions in the sequence to move forward on the ERAp reachability graph.
- No other way to select the first pair of transitions from the sequence of process transitions exists because ($-m1_{1\text{-}2}$, $-m2_{2\text{-}1}$) is a type-2 ERAp transition pair (See Definition 5.6).

Therefore, the mathematical contradiction in Theorem 3.1 [Pe95] is not correct. □

Point 5.6 Global state (<1, 0>,<ε, ε>) is a stable global state in protocol Π1, which is reachable from the initial global state by a sequence of process transitions ($-m1_{1\text{-}2}$, $-m2_{2\text{-}1}$, $-m3_{1\text{-}2}$, $+m1_{1\text{-}2}$, $+m2_{2\text{-}1}$, $+m3_{1\text{-}2}$). However, that stable global state is not included in the ERAp reachability graph:

- If ($-m1_{1\text{-}2}$, $+m1_{1\text{-}2}$) is selected as the first pair of transitions, then the reached global state is (<1, 2>,<ε, ε>). The second pair of transitions from the remaining process transitions in the sequence must be ($-m3_{1\text{-}2}$, $+m3_{1\text{-}2}$) because process transition $-m2_{2\text{-}1}$ is not executable at (<1, 2>,<ε, ε>). Then, the reached global state is (<2, 0>,<ε, ε>). The last pair of transitions from

the remaining process transitions in the sequence must be ($-m2_{2\text{-}1}$, $+m2_{2\text{-}1}$) that results in global state (<1, 1>,<ε, ε>).

- If ($-m2_{2\text{-}1}$, $+m2_{2\text{-}1}$) is selected as the first pair of transitions, then the current global state is (<1, 1>,<ε, ε>). Only process transition $-m3_{1\text{-}2}$ is executable at this global state, which leads to a blocking state (<2, 1>,<m3, ε>). Then, the second pair of transitions from the remaining process transitions in the sequence cannot be selected.

- No other way to select the first pair of transitions from the sequence of process transitions exists because ($-m1_{1\text{-}2}$, $-m2_{2\text{-}1}$) is a type-2 ERAp transition pair (See Definition 5.6).

Therefore, not every stable global state in a protocol is reachable by the ERAp strategy. In other words, $\Gamma_{STABLE} \not\subset \Gamma_{ERAp}$. □

Point 5.7 There are two deadlock states (<2, 2>,<ε, ε>) and (<2, 1>,<ε, ε>) in protocol Π1 as shown in Figure 3.2. However, none of them is included in the ERAp reachability graph. Therefore, deadlock freedom of a protocol is not verifiable by the ERAp strategy. □

In summary, we identified the following flaws in [Pe95]:

- The definition of transition pairs is not consistent with the use of them throughout the paper as pointed out in the redefinition of type-1 ERAp transition pairs (Definition 5.3).

- Although the ERAp strategy is claimed to be applicable to the general class of *n*-process protocols, the examples in the paper are restricted to the cyclic protocols.

- The mathematical support for the claims made in the paper does not follow the logic norms as pointed out in Point 5.2 and Point 5.4.
- Not every stable global state of a protocol is reachable by the ERAp strategy (Point 5.6).
- The ERAp strategy is not an extension of fair reachability analysis.
- The ERAp strategy fails in verifying deadlock freedom of a protocol as pointed out in Point 5.3 and Point 5.7.

The remaining question is the possibility of a relief strategy that can preserve the even total channel length property in the generated global state space to verify deadlock freedom of an n-process protocol with arbitrary topology. The following chapter will focus on the answer to that question.

Chapter 6 The ERAf Strategy

Recall that, the global state space Γ_{ERAp} generated by using the selected ERAp transition pairs is a proper subset of the even reachable global state space, i.e., $\Gamma_{ERAp} \subset \Gamma_{EVEN}$. The main failure of the ERAp strategy is that Γ_{ERAp} cannot include the stable global state space, Γ_{STABLE}, as attempted. Since $\Gamma_{STABLE} \subset \Gamma_{EVEN}$, ERAp can be saved, at the first thought, if the even reachable global state space is included in the generated global state space, i.e., $\Gamma_{EVEN} \subset \Gamma_{ERAp}$. A strategy is developed based on these ideas, which is called the *ERAf strategy* that stands for a full coverage of the even reachable global state space. Of course, such a name makes sense only when reachability graphs are finite. Indeed, this is the assumption since the beginning of this thesis.

The ERAf strategy hence must define an extended set of transition pairs for those purposes. Empirical results from the RELIEF tool will eventually judge the performance of ERAf, but at this point, one-half reduction in generated global states can be expected from the strategy. ERAf is still for the general class of n-process protocols. Since the *ERAf global state space* will be the even reachable global state space, $\Gamma_{ERAf} = \Gamma_{EVEN}$, then the even total channel length property will obviously hold for every generated global state. However, ERAf will not be a generalization of the classical fair reachability analysis because the ERAf reachability graph will not be always equivalent to the fair reachability graph of a protocol in the special case of two-process protocols.

The sections in this chapter are arranged as follows: Section 6.1 defines the ERAf preliminaries, Section 6.2 proves that ERAf can verify deadlock freedom of any n-process protocol with arbitrary topology, and Section 6.3 reports the performance of ERAf.

6.1 The ERAf Preliminaries

6.1.1 The ERAf Transition Pairs

Definition 6.1 A *type-1 ERAf transition pair* at a global state G_k, which is denoted by τ^1, is a pair of two process transitions constructed by one of the following conditions:

1. $(\exists i, j \in I \wedge i \neq j \wedge \sigma_{ij}, \rho_{ij} \in exec(G_k)) \Rightarrow \tau^1 = (\sigma_{ij}, \rho_{ij})$.
2. $(\exists i, j \in I \wedge i \neq j \wedge \sigma_{ij} \in exec(G_k) \wedge \rho_{ij} \in potent(G_k)) \Rightarrow \tau^1 = (\sigma_{ij}, \rho_{ij})$.
3. $(\exists i, j \in I \wedge i \neq j \wedge \rho_{ij} \in exec(G_k) \wedge \sigma_{ij} \in potent(G_k)) \Rightarrow \tau^1 = (\rho_{ij}, \sigma_{ij})$. □

The definitions of σ_{ij}, ρ_{ij} are already provided in Definition 2.2. The order of the transitions in a pair is not a matter in Condition 1, but it is a matter in Conditions 2 and 3. Condition 3 is for the case of potentially executable sending transitions, which is necessary to be explicit when the channels are bounded.

Type-1 ERAf transition pairs express the cases of two opposite actions from two different processes on the same channel. Moreover, since τ^1 always consists of two opposite actions involving the same channel, then the execution of τ^1 does not change the length of the involved channel. Also notice that, the single-channel pairs in the van der Schoot-Ural strategy of fair reachability analysis (Section 3.2.1.6) have exactly the same construction as type-1 ERAf transition pairs here.

Definition 6.2 A *type-2 ERAf transition pair* at a global state G_k, which is denoted by τ^2, is an unordered pair of two process transitions constructed by one of the following conditions:

1. $(\exists i, j, p, q \in I \wedge i \neq p \wedge j \neq q \wedge i \neq j \wedge \sigma_{ip}, \rho_{qj} \in exec(G_k)) \Rightarrow \tau^2 = (\sigma_{ip}, \rho_{qj})$.
2. $(\exists i, j, p, q \in I \wedge i \neq p \wedge j \neq q \wedge i \neq j \wedge \sigma_{ip}, \sigma_{jq} \in exec(G_k)) \Rightarrow \tau^2 = (\sigma_{ip}, \sigma_{jq})$.
3. $(\exists i, j, p, q \in I \wedge i \neq p \wedge j \neq q \wedge i \neq j \wedge \rho_{pi}, \rho_{qj} \in exec(G_k)) \Rightarrow \tau^2 = (\rho_{pi}, \rho_{qj})$.

Notice that, the pairs of two receptions from two different processes, which are missing in type-2 ERAp transition pairs, are now added to type-2 ERAf transition pairs. Type-2 ERAf transition pairs express the cases of two actions from two different processes on two different channels. The execution of τ^2 changes the length of each involved channel, but it preserves the even total channel length over all. In the special case of two-process protocols, type-2 ERAf transition pairs drop Condition 1 in the above construction and remain the same as the ring tuples in the van der Schoot-Ural strategy of fair reachability analysis (Section 3.2.1.6). □

Definition 6.3 A *type-3 ERAf transition pair* at a global state G_k, which is denoted by τ^3, is an ordered pair of two process transitions constructed by one of the following conditions:

1. $(\exists i, j \in I \wedge i \neq j \wedge \sigma_{ij} \in exec(G_k) \wedge \exists G_l \in \Gamma\ G_k \text{—}\sigma_{ij}\text{→} G_l \wedge \exists p \in I \wedge i \neq p \wedge \sigma_{ip} \in exec(G_l)) \Rightarrow \tau^3 = (\sigma_{ij}, \sigma_{ip})$.

2. $(\exists i, j \in I \wedge i \neq j \wedge \sigma_{ij} \in exec(G_k) \wedge \exists G_l \in \Gamma\ G_k \text{—}\sigma_{ij}\text{→} G_l \wedge \exists p \in I \wedge i \neq p \wedge \rho_{pi} \in exec(G_l)) \Rightarrow \tau^3 = (\sigma_{ij}, \rho_{pi})$.

3. $(\exists i, j \in I \wedge i \neq j \wedge \rho_{ji} \in exec(G_k) \wedge \exists G_l \in \Gamma\ G_k \text{—}\rho_{ji}\text{→} G_l \wedge \exists p \in I \wedge i \neq p \wedge \sigma_{ip} \in exec(G_l)) \Rightarrow \tau^3 = (\rho_{ji}, \sigma_{ip})$.

4. $(\exists i, j \in I \wedge i \neq j \wedge \rho_{ji} \in exec(G_k) \wedge \exists G_l \in \Gamma\ G_k \text{—}\rho_{ji}\text{→} G_l \wedge \exists p \in I \wedge i \neq p \wedge \rho_{pi} \in exec(G_l)) \Rightarrow \tau^3 = (\rho_{ji}, \rho_{pi})$.

Notice that G_l is a sequential immediate successor of G_k as defined in Definition 2.7. Type-3 ERAf transition pairs are the same as those of ERAp strategy and express the cases of two consecutive transitions from a single process. The fair progress property of the classical fair reachability analysis [GoHa85] is lost from ERAf since then. □

Definition 6.4 An executable ERAf transition pair at a global state G_k is either a type-1, type-2, or type-3 ERAf transition pair at G_k. The set of all executable ERAf transition pairs at G_k is denoted by $exec_{ERAf}(G_k)$. □

Notice that the execution of an executable ERAf transition pair preserves the even total channel length property. Conceptually, $exec_{ERAf}(G_k)$ in the ERAf strategy is the union of all three types of transition pairs while $select_{ERAp}(G_k)$ in the ERAp strategy is the ordered and preclusive selection from the three types of transition pairs (Definition 5.6).

6.1.2 The ERAf Relations

In the ERAf reachability analysis of a protocol, a global state, say G_k, generates another global state, say G_l, if there exists an executable ERAf transition pair τ at G_k, and the two process transitions in that pair are executed to reach G_l. Notice that in the ERAf relations $\tau \in exec_{ERAf}(G_k)$ while $\tau \in select_{ERAp}(G_k)$ in the ERAp relations (Section 5.1.3).

Definition 6.5 A global state G_l is an *ERAf immediate successor* of a global state G_k, denoted by $G_k \rightarrow_{ERAf} G_l$ or $G_k \text{—} \tau \rightarrow_{ERAf} G_l$, iff $\exists \tau \in exec_{ERAf}(G_k)$ such that $G_k \text{—} \tau \rightarrow^* G_l$. □

Notice that in Definition 6.5, the ERAf immediate successor defines a binary relation between the reachable global states, which is denoted by $\rightarrow_{ERAf}$.

Definition 6.6 Let $\rightarrow_{ERAf}^*$ be the reflexive and transitive closure of $\rightarrow_{ERAf}$.

1. G_m is a reachable global state by ERAf from G_k (or an ERAf successor of G_k) iff $G_k \rightarrow_{ERAf}^* G_m$.

2. G_m is a reachable global state by ERAf iff $G_0 \rightarrow_{ERAf}^* G_m$.

$G_k \text{—} \tau_1 \rightarrow_{ERAf} G_{k(1)} \wedge G_{k(1)} \text{—} \tau_2 \rightarrow_{ERAf} G_{k(2)} \wedge \ldots \wedge G_{k(j-1)} \text{—} \tau_j \rightarrow_{ERAf} G_m$ is also denoted by $G_k \text{—} \tau_1 \tau_2 \ldots \tau_j \rightarrow_{ERAf}^* G_m$. □

Reachable global state space Γ_{ERAf} of a protocol by ERAf can be represented by a directed graph, called *ERAf reachability graph* and also denoted by Γ_{ERAf}, in which the nodes represent global states and the arcs stand for the ERAf immediate successor relation between global states, which will be also called global transitions in this thesis.

6.2 ERAf and Deadlock Detection

Lemma 6.1 A global state reachable by ERAf is sequentially reachable.

Proof: Suppose that G_k is a reachable global state by ERAf,

$\Rightarrow G_0 \rightarrow_{ERAf}{}^* G_k$ (by Definition 6.6)

$\Rightarrow \exists(\tau_1 \tau_2 ... \tau_j)\ G_0 \text{—} \tau_1 \tau_2 ... \tau_j \rightarrow_{ERAf}{}^* G_k$

$\Rightarrow \exists(t_1^1, t_1^2, t_2^1, t_2^2, ..., t_j^1, t_j^2)\ \tau_1=(t_1^1, t_1^2),\ \tau_2=(t_2^1, t_2^2),\ ...,\ \tau_j=(t_j^1, t_j^2)$

(by Definition 6.4).

$\Rightarrow \exists(t_1^1 t_1^2 t_2^1 t_2^2 ... t_j^1 t_j^2)\ G_0 \text{—} t_1^1 t_1^2 t_2^1 t_2^2 ... t_j^1 t_j^2 \rightarrow {}^* G_k$

$\Rightarrow G_0 \rightarrow {}^* G_k$

then, G_k is a sequentially reachable global state by Definition 2.8. □

Lemma 6.2 If a global state G_m is sequentially reachable from another global state G_k by any two process transitions, then G_m is an ERAf immediate successor of G_k by the ERAf transition pair composed of the two process transitions in order.

Proof: G_m is a sequentially reachable global state from G_k by two process transitions,

$\Rightarrow \exists G_l \in \Gamma G_k \rightarrow G_l \rightarrow G_m$ (by Definition 2.8)

$\Rightarrow \exists t_1 \in exec(G_k),\ \exists t_2 \in exec(G_l)\ G_k \text{—} t_1 \rightarrow G_l \wedge G_l \text{—} t_2 \rightarrow G_m$ (by Definition 2.9)

$\Rightarrow \exists \tau \in exec_{ERAf}(G_k)\ \tau=(t_1, t_2)$ (by Definition 6.4)

$\Rightarrow G_k \text{—} \tau \rightarrow {}^* G_m$ (by Definition 2.8)

then, G_m is an ERAf immediate successor of G_k by Definition 6.5. □

Theorem 6.1 An even reachable global state is reachable by ERAf.

Proof: Suppose that G_k is an even reachable global state that is sequentially reachable by executing $2(k \geq 0)$-transition steps, i.e.,

$\Rightarrow G_0 \rightarrow {}^* G_k$ (by Definition 2.8)

$\Rightarrow \exists(t_1^1 t_1^2 t_2^1 t_2^2 ... t_k^1 t_k^2)\ G_0 \text{—} t_1^1 t_1^2 t_2^1 t_2^2 ... t_k^1 t_k^2 \rightarrow^* G_k$

$\Rightarrow \exists(\tau_1 \tau_2 ... \tau_k)\ \tau_1=(t_1^1, t_1^2),\ \tau_2=(t_2^1, t_2^2),\ ...,\ \tau_k=(t_k^1, t_k^2)$ (by Definition 6.4)

$\Rightarrow \exists(\tau_1 \tau_2 ... \tau_k)\ G_0 \text{—} \tau_1 \tau_2 ... \tau_k \rightarrow_{ERAf}^* G_k$ (by Lemma 6.2)

$\Rightarrow G_0 \rightarrow_{ERAf}^* G_k$

then, G_k is a reachable global state by ERAf by Definition 6.6. □

Corollary 6.1 A protocol is deadlock free iff every stable global state that is reachable by ERAf is free from deadlock.

Proof: By Definition 5.2, a deadlock state is a stable global state and a stable global state is an even reachable global state. Moreover, by Theorem 6.1, every even reachable global state is reachable by ERAf. Hence, deadlock freedom is verifiable by the ERAf strategy. □

By Lemma 6.1, a global state reachable by the ERAf strategy is sequentially reachable. Furthermore, according to Section 6.1, the total length of all the channels in a global state reachable by ERAf is even, then it is an even reachable global state (Definition 5.1). In reverse, by Theorem 6.1, an even reachable global state is reachable by ERAf. Therefore, the ERAf global state space is simply the even reachable global state space as illustrated in Figure 6.1. Proposition 6.1 about global state reduction is derived from that view.

Proposition 6.1 One-half reduction in the number of generated global states is expected from the ERAf strategy. □

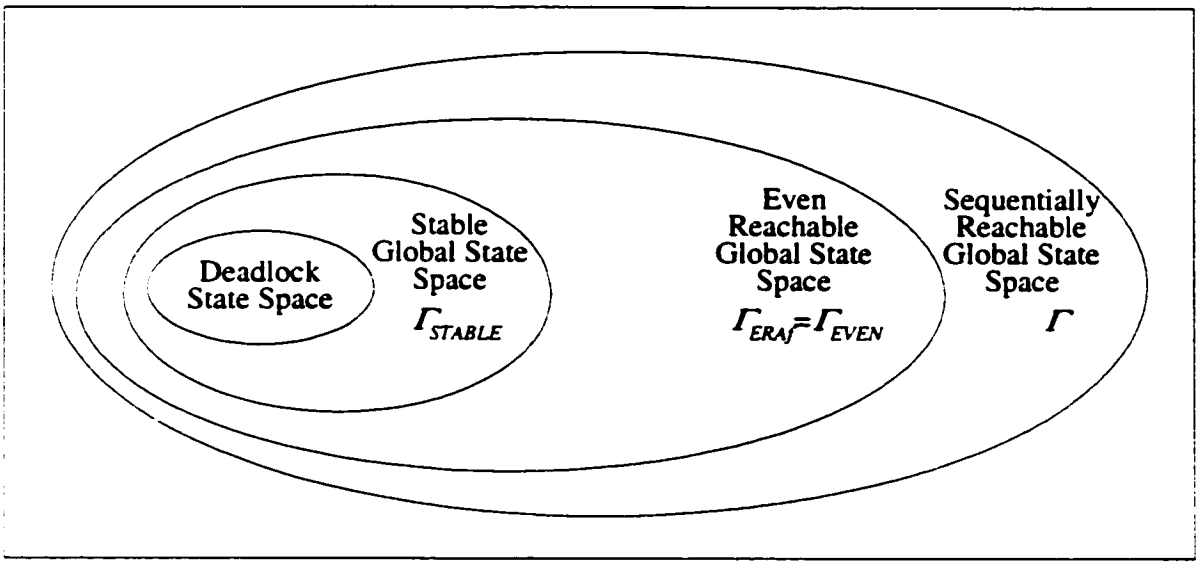

Figure 6.1 The reachable global state space by ERAf

The ERAf reachability graph of two-process protocol Π1 is given as an example in Figure 6.2. Again, the sequentially reachable global states that are not reachable by the ERAf strategy are left blank in place for comparison purposes. Suppose that, the sequentially reachable global states are arranged in the graph such that those generated by the same number of transition steps are on the same row, then all global states on the even-step rows remain the ERAf reachability graph while all others on the odd-step rows are gone.

The ERAf results reported by RELIEF is also given in Figure 6.3. Notice that, in the special case of two-process protocols, every fair reachable global state is reachable by ERAf, but the reverse is not always true. It is not difficult to prove that inclusion, however, the fair reachability graph of the same protocol is already given in Figure 3.3 for reference purposes. In this particular example, ERAf generates five more global states beside the fair reachable global state space due to the additional use of type-3 ERAf transition pairs.

By Corollary 6.1, it is clear that the ERAf strategy is effective in verifying the absence of deadlocks. In Section 6.3, the performance of ERAf will be studied.

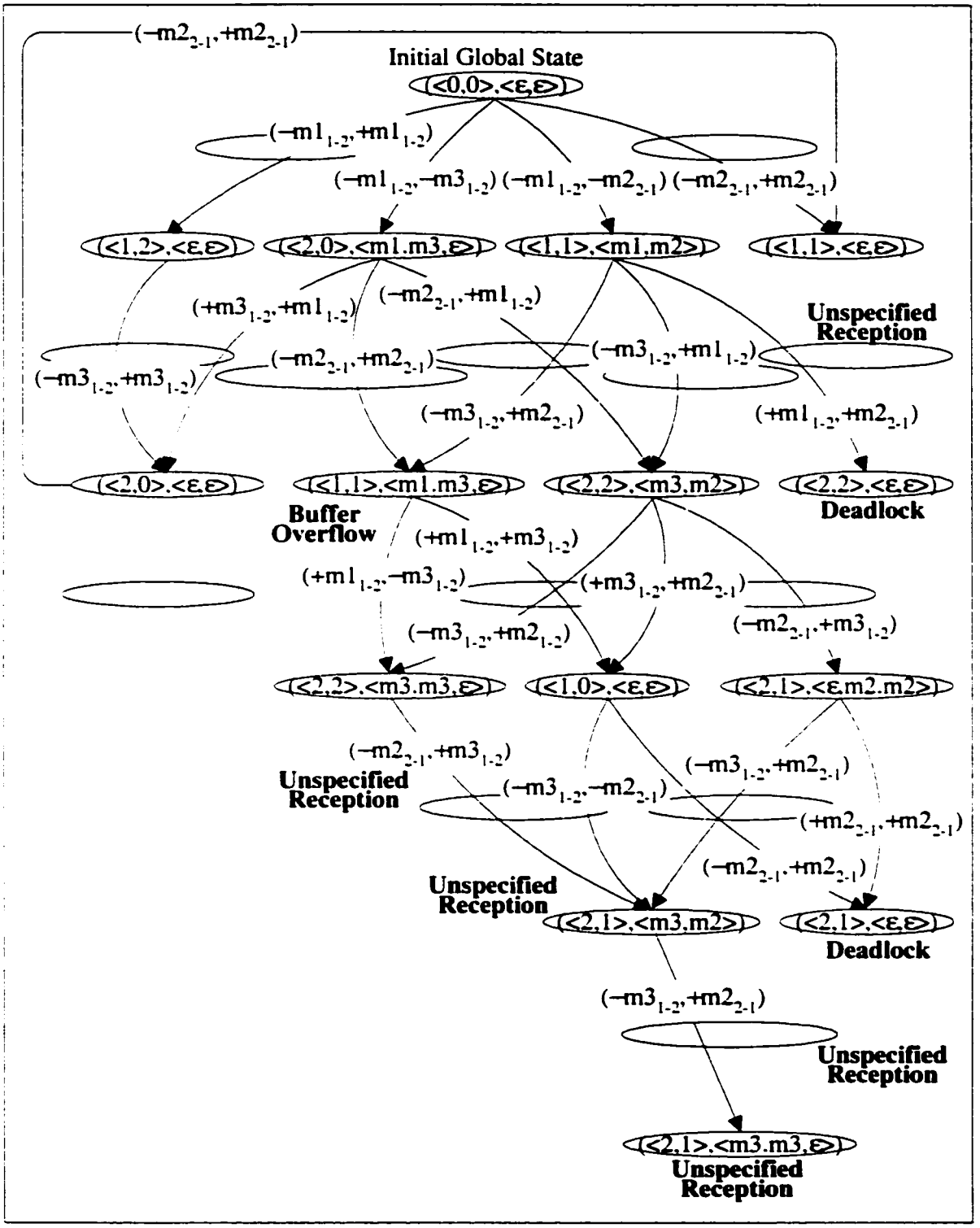

Figure 6.2 The ERAf reachability graph of two-process protocol Π1

```
**********************************************************************
                    EVEN REACHABILITY ANALYSIS (ERA)

       Full coverage.....................................(f)
       Peng coverage.....................................(p)


       File Utilities....................................(u)
       Back to Previous Menu.............................(b)
       Main Menu.........................................(m)
       Exit..............................................(x)

       Choice............................................(f

Analyzing Protocol No. 1, wait...done

Read report (y/n)? y

EVEN REACHABILITY ANALYSIS (ERAf)
Deadlock Detector reports:

PROTOCOL No.: 1
NUMBER OF PROCESSES: 2
Process No. 1: States Nos.: 0 1 2 3
Process No. 2: States Nos.: 0 1 2 3
NUMBER OF  CHANNELS: 2
Channel Bound: 2
Channel  Nos.: 1-2 2-1

  BLOCKING STATES:
  **********************************************************************
   <State i,State j,...>,<Queue i-j,....,Queue j-i,...>
  **********************************************************************
   <2,1>,<,>
   <2,1>,<m3.m3,>
   <2,2>,<,>

  UNSPECIFIED RECEPTION CAUSING (PROCESS STATE,MESSAGE) PAIRS:
  **********************************************************************
          Process No.      State            Message     (from) Process No.
  **********************************************************************
          2                1                m3                 1

  BUFFER OVERFLOW CAUSING (PROCESS STATE,MESSAGE) PAIRS:
  **********************************************************************
          Process No.      State            Message       (to) Process No.
  **********************************************************************
          1                1                m3                 2

  NUMBER OF PROTOCOL DESIGN ERRORS:
  **********************************************************************
      Blocking     Deadlock     UR Causing     BO Causing     Non-Executable
      States       States       (process,M)    (process,M)    Transitions
                                Pairs          Pairs
  **********************************************************************
      >= 3         2            >= 1           >= 1           ?

  Global States      : 15
  Global Transitions: 23
  Space Consumed     : 0.75 (KBytes)
  Time Consumed      : 0.00 (Seconds)
```

Figure 6.3 The ERAf results reported by RELIEF for two-process protocol Π1

6.3 The Performance of ERAf

6.3.1 The ERAf Empirical Results

The empirical results for ERAf compared with CRA by RELIEF using the set of four-hundred APS protocols (Section 4.2.1) are transferred to Table 6.1 in the typical form described in Section 4.2.2. From the corresponding empirical study, all deadlocks in the four-hundred protocols, which are revealed by CRA are also revealed by the ERAf strategy.

- One-half reduction in the number of stored global states compared with conventional reachability analysis is just as expected in Proposition 6.1. That reduction is likely constant for any n-process protocols and for any concurrency levels.

- In terms of the number of processes, the reduction in generated global transitions is decreasing from 13.78 per cent in the average for two-process protocols down to –52.03 per cent in the average for eight-process protocols. In terms of the concurrency levels, the reduction in generated global transitions is decreasing from 10.19 per cent in the average when the concurrency levels are in the [0,1] interval down to –55.43 per cent in the average when the concurrency levels are in the (3,4] interval. An analytic explanation for these negative reduction is provided in Section 6.3.2.

***n*-process protocols**							
***n*=**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Reduction in global states (%)							
Average	51.18	50.38	50.10	50.06	50.02	50.00	50.00
Reduction in global transitions (%)							
Average	13.78	-1.05	-14.78	-12.64	-28.19	-37.18	-52.03
Reduction in memory space (%)							
Average	51.03	50.32	50.08	50.04	50.02	50.00	50.00

Reduction in execution time (%)							
Average	0.79	-38.92	-76.47	-98.40	-156.03	-198.97	-273.18

Levels of concurrency	**[0, 1]**	**(1, 2]**	**(2, 3]**	**(3, 4]**	**(4, 5]**
Reduction in global states (%)					
Average	50.94	50.13	50.02	50.00	
Reduction in global transitions (%)					
Average	10.19	-15.11	-40.73	-55.43	
Reduction in memory space (%)					
Average	50.82	50.11	50.01	50.00	
Reduction in execution time (%)					
Average	-16.42	-98.01	-206.78	-272.64	

Table 6.1 The performance of ERAf compared with CRA by RELIEF

Compared with the empirical results in Table 4.5 for LRAd using the same set of four-hundred APS protocols, the performance of ERAf appears inferior indeed.

6.3.2 An Analytic Explanation

Property 6.1 The maximum number of distinct process transitions that can lead to a sequentially reachable global state of a protocol is the maximum number of communication channels in the protocol.

Proof: A sequentially reachable global state is reached by a transition only when that transition causes a single change in the content of a channel. Suppose there are two transitions that lead to a global state by making changes on the same channel, then this results in an additional global state by Definition 2.7 and raises a contradiction. For a protocol with n processes, $n \geq 2$, the maximum number of communication channels is $n(n-1)$. Then, the maximum number of distinct transitions leading to a sequentially reachable global state is also $n(n-1)$. □

Of course, the number of distinct process transitions leading to a sequentially reachable global state is lower when only a subset of channels are active, i.e., $active(C) \subset C$.

Property 6.2 The maximum number of distinct process transitions executed at a sequentially reachable global state of a protocol is $n(n-1)(m+1)$, where m is the maximum number of distinct messages sent from one process to another and n is the number of processes in the protocol.

Proof: Notice that the number of distinct process transitions executed at a sequentially reachable global state G_k of a protocol is the cardinality of the set of executable transitions of the protocol at G_k. The construction of the executable transition set, $exec(G_k)$, is provided in Definition 2.9.

In a protocol, a process can have $n-1$ incoming channels and $n-1$ outgoing channels. The maximum number of transitions that a process can make at a global state by sending messages is $m(n-1)$. On the other hand, the maximum number of transitions that a process can make at a global state by receiving messages is $n-1$. Then the maximum number of transitions that a process can make at a global state is $(n-1)(m+1)$. Since there are n processes, then the maximum number of process transitions executed at a sequentially reachable global state is $n(n-1)(m+1)$. □

Property 6.3 The maximum number of ERAf transition pairs constructed from the process transitions at a sequentially reachable global state of a protocol is $n^2(n-1)^2(m+1)$, where n is the number of processes in the protocol and m is the maximum number of distinct messages sent from one process to another.

Proof: Notice that the construction of ERAf transition pairs exhausts all possible combination of two process transitions. The maximum number of ERAf transition pairs constructed from the process transitions at a sequential global state of a protocol is the product of the maximum number of transitions leading to and executed at that global state, which are presented in Property 6.1 and Property 6.2, respectively. □

Figure 6.4 depicts a global state that is sequentially reachable from the initial global state of a protocol by an odd number of transition steps. Suppose that the global state has $n(n-1)$ incoming transitions and $n(n-1)(m+1)$ outgoing transitions. The number of transitions by the conventional reachability analysis related to this global state is $n(n-1)+n(n-1)(m+1)=n(n-1)(m+2)$. By Property 6.3, the maximum number of resulting ERAf transition pairs is $n^2(n-1)^2(m+1)$. When n grows, $n^2(n-1)^2(m+1)$ becomes much greater than $n(n-1)(m+2)$ causing an explosion in global transitions by the ERAf strategy.

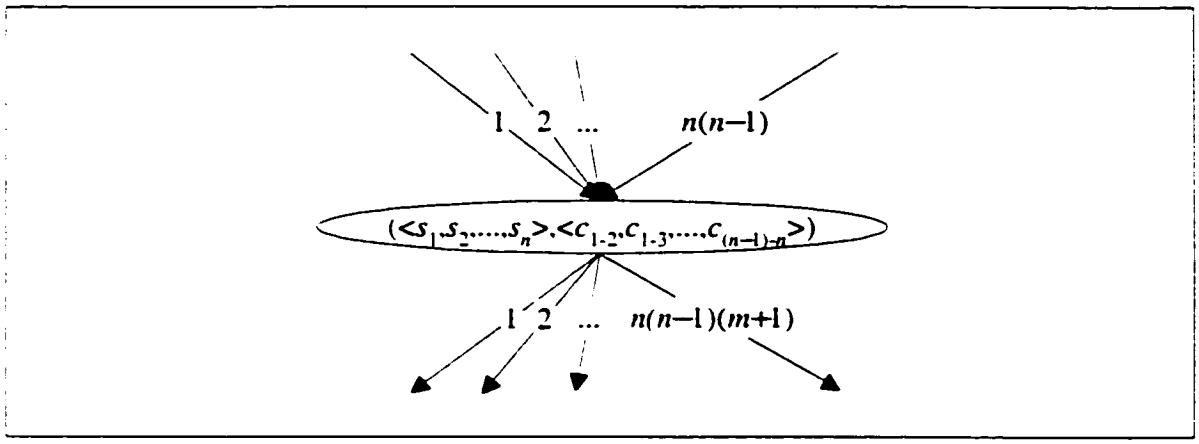

Figure 6.4 Incoming and outgoing transitions to and from a sequential global state

In fact, the unordered pairs in ERAf transition pairs can help reduce the maximum number given in the above formula. However, not every ERAf transition pairs is unordered and that fact cannot save the global transition explosion eventually. As an example, consider the case when n=5 and m=2. At a global state, the maximum number of incoming transitions is 5(5−1)=20 and the maximum number of outgoing transitions is 5(5−1)(2+1)=60. Then, the number of sequential transitions is 80 and the number of ERAf transition pairs is 1200. Even if all ERAf transition pairs were unordered, the number of ERAf transition pairs would be 600 and still greater than the number of sequential transitions. This calculation shows that the negative reduction in the number of global transitions in Table 6.1 is not unexpected.

6.4 The Refined Transition Pairs

So far, the results of the experiment suggest that, there are possible redundancies in the ERAf transition pairs in exploring the even reachable global state space of n-process protocols. Therefore, ERAf transition pairs are visited again to find out whether a reduction can be made for them. There are two ideas for the refinement:

- Type-1 and type-2 transition pairs should remain to guarantee that, even reachability graph reduces to fair reachability graph when the number of processes in a protocol is equal to two. Although there is no mathematical proof showing that fair reachability analysis is an optimal strategy for two-process protocols, losing the fair reachability graph may repeat the failure of ERAp as pointed out in Section 5.4.

- Type-3 transition pairs are used at a global state only when there exist neither type-1 nor type-2 transitions at that global state. That means, in this refinement, the existence of type-1 or type-2 transition pairs will preclude the use of type-3 transition pairs.

An experiment is carried out for such a conservative refinement in transition pairs. Unfortunately, RELIEF shows that some deadlock states in some APS protocols are left undetected by the experimental strategy.

FRA is not applicable to the general class of n-process protocols with arbitrary topology. SRA and LRA require delicate algorithms to establish the simultaneously executable sets. The ideas of even reachability analysis open a vision on using simple activities to explore reachable global states of n-process protocols with arbitrary topology. Each transition pair is composed of only two process transitions. The construction is straightforward, but the applicability is large compared with both simultaneous reachability analysis and fair reachability analysis. Learning the failure of the even reachability analysis proposed by Peng, a set of extended transition pairs is

successful in verifying deadlock freedom of a protocol. Around fifty per cent reduction in generated global states is achieved, however, a trade-off appears as the negative reduction in the number of global transitions generated by the ERAf strategy.

Chapter 7 Concluding Remarks

Over the last two decades, many relief strategies have been proposed to relieve the global state explosion problem in the conventional reachability analysis. One of the most popular strategies is fair reachability analysis. Starting with the Rubin-West strategy [RuWe82] for the class of two-process protocols, fair reachability analysis results in a large amount of reduction in global state generation against the conventional reachability analysis. The activity is rather simple by keeping the two processes to progress at the same speed during state exploration, which is called fair progress property. This strategy generates and analyzes only the global states that are reached by executing equal number of transitions in each process, i.e., by executing a pair of transitions at a time, one per process.

The Rubin-West strategy inspires a number of generalized fair reachability analyses. The Liu-Miller strategy generalizes the fair progress property to the equal channel length property [LiMi94a]. Then, the Liu-Miller fair reachability analysis can apply to the class of n-process cyclic protocols. The generalization road continues with the van der Schoot-Ural strategy. The equal channel length property in the Liu-Miller strategy is generalized to the ring equilibrium property [ScUr95b]. The van der Schoot-Ural fair reachability analysis can apply to the class of n-process multi-cyclic protocols. The amount of reduction in global state generation of fair reachability analysis in verifying deadlock freedom of a protocol is always exciting. However, Liu, Miller, van der Schoot, and Ural together argue that the strategy of fair reachability analysis cannot be extended beyond multi-cyclic protocols [LiMiScUr96]. A fair reachability analysis for the general class of n-process protocols with arbitrary topology remains open.

On the other hand, simultaneous reachability analysis [ÖzUr95] focuses on the ideas of simultaneous execution of transitions. Simultaneous reachability analysis can apply to the general class of n-process protocols without any restrictions on the topology of the protocol or the structure of the processes. By replacing $k!$ interleaving of process transitions by a simultaneous executable transition set, simultaneous reachability analysis makes significant reduction in verifying deadlock freedom of a protocol. However, simultaneous reachability analysis or the improved version, which is called leaping reachability analysis [ScUr96b], require a delicate algorithm to select the simultaneous executable transition sets [ÖzUr95].

The ideas of even reachability analysis [Pe95] open a vision on using simple activities to explore reachable global states of n-process protocols with arbitrary topology. Each transition pair is composed of only two process transitions. The construction is straightforward but the applicability is large compared with both simultaneous reachability analysis and fair reachability analysis. Unfortunately, the selected transition pairs proposed in [Pe95] are not sufficient to detect all deadlock states in a protocol.

Learning the failure of the even reachability analysis proposed by Peng, a set of extended transition pairs is used in a new version of even reachability analysis. While the selected transition pairs generate only a proper subset of the even reachable global state space, the extended set of transition pairs generates the entire even reachable global state space in a finite global state space. Around fifty per cent reduction in generated global states is achieved, however, a trade-off appears as the negative reduction in the number of global transitions generated by the new strategy.

As an attempt to reduce the possible redundancies in global transitions by the above even reachability analysis, a reduced version of the extended set of transition pairs is experimented. Unfortunately, the RELIEF tool shows that some deadlock states in some APS protocols are left undetected by the experimental strategy.

Chapter 8 References

[BeHo89] F. Belina, D. Hogrefe, "The CCITT Specification and Description Language SDL." *Computer Networks and ISDN Systems*, Vol. 16, pp. 311-341, 1989.

[Bo78] G.v. Bochmann, "Finite State Description of Communication Protocols." *Computer Networks*, Vol. 2, pp. 361-372, 1978.

[BrZa83] D. Brand, P. Zafiropulo, "On Communicating Finite State Machines." *ACM Transactions on Programming Languages and Systems*, Vol. 30, No. 2, A, pp. 323-342, 1983.

[ChMi83] T.Y. Choi, R.H. Miller, "A Decomposition Method for the Analysis and Design of Finite State Protocols." *ACM SIGOMM'83*, pp. 167-176, 1983.

[CoLeRi90] T.H. Cormen, C.E. Leiserson, R.L. Rivest, "Introduction to Algorithms." Second Edition, The MIT Press, 1990.

[CoPeAy85] J.P. Courtiat, A. Pedroza, J.M. Ayache, "A Simulation Environment for Protocol Specifications in Estelle." *In Proceedings of the 5th International Workshop on Protocol Specification, Testing and Verification*, Amsterdam, North Holland, 1986.

[Da80] A.A.S. Danthine, "Protocol Representation with Finite-state Models." *IEEE Transactions on Communications*, Vol. COM-28, pp. 632-643, April 1980.

[Fi88] A. Finkel, "A New Class of Analyzable CFSMs with Unbounded FIFO Channels." *Protocol Specification, Testing and Verification*, VIII, pp. 283-294, 1988.

[GoHa85] M.G. Gouda, J.Y. Han, "Protocol Validation by Fair Progress State Exploration." *Computer Networks and ISDN Systems*, Vol. 9, pp. 353-361, North Holland, 1985.

[GoYu84] M.G. Gouda, Y.T. Yu, "Protocol Validation by Maximal Progress State Exploration." *IEEE Transactions on Communications*, Vol. 32, No. 1, pp. 94-97, 1984.

[Ho88] G.J. Holzmann, "An Improved Protocol Reachability Analysis Technique." *Software Practice and Experience*, Vol. 9, pp. 137-161, 1988.

[Ho90] G.J. Holzmann, "Algorithm for Automatic Protocol Verification." *AT&T Technical Journal*, Vol. 69, No. 2, pp. 32-44, 1990.

[Ho92] G.J. Holzmann, *Design and Validation of Computer Protocols*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1992.

[ItIc83] M. Itoh, H. Ichikawa, "Protocol Verification Using Reduced Reachability Analysis." *The Transactions on the IECE of Japan*, Vol. E66, No. 2, pp. 137-161, 1983.

[Li85] R.Jr. Linn, "The Features and Facilities of Estelle." *In Proceedings of the 5th International Workshop on Protocol Specification, Testing and Verification*, June 1985.

[LiChLi87] F.J. Lin, P.M. Chu, M.T. Liu, "Protocol Verification Using Reachability Analysis: The State Explosion Problem and Relief Strategy." *Computer Communication Review*, Vol. 17, No. 5, 1987.

[LiMi93] H. Liu, R.E. Miller, "Deadlock Detection for Cyclic protocols Using Generalized Fair Reachability Analysis." *Technical Report*, Department of Computer Science, University of Maryland at College Park, CS-TR-3135, September 1993.

[LiMi94a] H. Liu, R.E. Miller, "Generalized Fair Reachability Analysis for Cyclic Protocols: Part 1." *Protocol Specification, Testing and Verification*,

XIV, Elsevier Science Publishers B.V., pp. 271-286, North-Holland, 1994.

[LiMi94b] H. Liu, R.E. Miller, "Generalized Fair Reachability Analysis for Cyclic Protocols: Decidability for Logical Correctness Problems." *In Proceedings of the International Conference on Network Protocols*, Boston, MA, 1994.

[LiMi96] H. Liu, R.E. Miller, "Generalized Fair Reachability Analysis for Cyclic Protocols." *IEEE/ACM Transactions on Networking*, Vol. 4, No. 2, pp. 192-204, 1996.

[LiMiScUr96] H. Liu, R.E. Miller, H. van der Schoot, H. Ural, "Deadlock Detection by Fair Reachability Analysis: From Cyclic to Multi-cyclic Protocols (and Beyond?)." *In Proceedings of the 16th IEEE International Conference on Distributed Computing Systems*, pp. 605-612, Hong Kong, May 1996.

[LoFaHa92] L. Logrippo, M. Faci, M. Haj-Hussein, "An Introduction to LOTOS: Learning by Examples." *Computer Networks and ISDN Systems*, Vol. 23, pp. 325-342, 1992.

[MaSa87] N.F. Maxemchuk, K.K. Sabnani, "Probabilistic Verification of Communication Protocols." *Protocol Specification, Testing and Verification*, VII, pp. 307-320, 1987.

[ÖzUr94] K. Özdemir, H. Ural, "Deadlock Detection in CFSM Models via Simultaneous Executable Sets." *In Proceedings of the 6th International Conference on Communications and Information*, pp. 673-688, Peterborough, Ontario, Canada, 1994.

[ÖzUr95] K. Özdemir, H. Ural, "Protocol Validation by Simultaneous Reachability Analysis." *Technical Report*, Department of Computer Science, University of Ottawa, Ontario, Canada, TR-95-09, March 1995.

[ÖzUr97] K. Özdemir, H. Ural, "Protocol Validation by Simultaneous Reachability Analysis." *Computer Communications*, 20, pp. 772-788, 1997.

[ÖzUr98] K. Özdemir, H. Ural, "Erratum to "Protocol Validation by Simultaneous Reachability Analysis"." *Computer Communications*, 21, p. 591, 1998.

[Öz95] K. Özdemir, "Verifying the Safety Properties of Concurrent Systems via Simultaneous Reachability", *Ph.D. Thesis*, Department of Computer Science, University of Ottawa, Ontario, Canada, 1995.

[Pa87] J. Pachl, "Protocol Description and Analysis based on a State Transition Model with Channel Expression." *Protocol Specification, Testing and Verification*, VII, pp. 307-320, 1987.

[Pe95] W. Peng, "Deadlock Detection in Communicating Finite State Machines by Even Reachability Analysis." *In Proceedings of the 4th International Conference on Computer Communications and Networks*, Las Vegas, September 1995.

[PrSa91] R. Probert, K. Saleh, "Synthesis of Communicating Protocols: Survey and Assessment." *IEEE Transactions on Computers*, Vol. 40, No. 4, April 1991.

[RuWe82] J. Rubin, C.H. West, "An Improved Protocol Validation Technique." *Computer Networks and ISDN Systems*, Vol. 6, pp. 65-73, 1982.

[ScUr95a] H. van der Schoot, H. Ural, "Deciding Deadlock-freedom of Daisy-chain Protocol by Fair Reachability Analysis." *Technical Report*, Department of Computer Science, University of Ottawa, Ontario, Canada, TR-95-10, March 1995.

[ScUr95b] H. van der Schoot, H. Ural, "Deadlock Detection by Fair Reachability Analysis: Multi-cyclic Protocols and Beyond." *Technical Report*, Department of Computer Science, University of Ottawa, Ontario, Canada, TR-95-17, July 1995.

[ScUr95c] H. van der Schoot, H. Ural, "On Generalizing Fair Reachability Analysis to Protocols with Arbitrary Topologies." *In Proceedings of the 14th ACM Symposium on Principles of Distributed Computing*, p. 267, Ottawa, Ontario, Canada, 1995.

[ScUr95d] H. van der Schoot, H. Ural, "On Improving Simultaneous Reachability Analysis for the Efficient Verification of Deadlock-freedom." *Technical Report*, Department of Computer Science, University of Ottawa, Ontario, Canada, TR-95-20, December 1995.

[ScUr96a] H. van der Schoot, H. Ural, "Deciding Deadlock-freedom of Daisy-chain Protocols by Fair Reachability Analysis." *In Proceedings of the 11th IEEE International Symposium on Computer and Information Sciences*, Antalya, Turkey, 1996.

[ScUr96b] H. van der Schoot, H. Ural, "Protocol Verification by Leaping Reachability Analysis." *In Proceedings of IEEE International Conference on Computer Communications and Networks*, Rockville, MD, pp. 334-339, October 1996.

[ScUr96c] H. van der Schoot, H. Ural, "A Uniform Approach to Tackle State Explosion in Verifying Progress Properties for Networks of CFSMs." *Technical Report*, Department of Computer Science, University of Ottawa, Ontario, Canada, TR-96-13, November 1996.

[ScUr97] H. van der Schoot, H. Ural, "Leaping Reachability Analysis: A Uniform and Property-driven Relief Strategy for Protocol Verification." *Technical Report*, Department of Computer Science, University of Ottawa, Ontario, Canada, TR-97, 1997.

[ScUr98a] H. van der Schoot, H. Ural, "On Improving Reachability Analysis for Verifying Progress Properties of Networks of CFSMs." *In Proceedings of the 18th IEEE International Conference on Distributed Computing Systems*, pp. 130-137, Amsterdam, The Netherlands, May 1998.

[ScUr98b] H. van der Schoot, H. Ural, "An Improvement of Partial-order Verification." *Software Testing, Verification and Reliability*, pp. 83-102, VIII, John Wiley & Sons Ltd., 1998.

[Si82] D.P. Sidfhu, "Protocol Design Rules." *Protocol Specification, Testing and Verification*, pp. 283-300, 1982.

[VuCo82] S.T. Vuong, D.D. Cowan, "A Decomposition Method for the Validation of Structured Protocols." *INFOCOM*, pp. 209-220, 1982.

[We86] C.H. West, "Protocol Verification by Random State Exploration." *Protocol Specification, Testing and Verification*, VI, pp. 233-242, 1986.

[Yu88] M.C. Yuang, "Survey of Protocol Verification Techniques based on Finite State Machine Models." *In Proceedings of the NBS Computer Networking Symposium*, pp. 164-172, 1988.

[YuGo82] Y.T. Yu, M.G. Gouda, "Deadlock Detection for a Class of Communicating Finite State Machines." *IEEE Transactions on Communications*, Vol. 30, No. 12, pp. 2514-2518, 1982.

[Za78] P. Zafiropulo, "Protocol Validation by Duologue-matrix Analysis." *IEEE Transactions on Communications*, Vol. COM-26, No. 8, pp. 1187-1194, 1978.

[ZaTaShNo88] Y.X. Zang, K. Takahashi, N. Shiratori, S. Noguchi, "An Interactive Synthesis Algorithm Using a Global State Transition Graph.", *IEEE Transactions on Software Engineering*, Vol. 14, No. 3, pp. 394-404, 1978.

[ZhBo86] J. Zhao, G.v. Bochmann, "Reduced Reachability Analysis of Communication Protocols: A New Approach." *Protocol Specification, Testing and Verification*, VI, pp. 243-254, 1986.