

CSE Centre

Computer Systems Engineering Centre
School of Electrical and Information Engineering
University of South Australia
AUSTRALIA

Language Analysis of the Class of Stop-and-Wait Protocols

Guy Edward Gallasch
Jonathan Billington

January 2008

Summary

The correct operation of computer protocols is essential to the smooth operation of the distributed systems that facilitate our global economy. Formal techniques provide our best chance to ensure that protocol designs are free from errors. This report presents a Coloured Petri net (CPN) model of the class of Stop-and-Wait protocols (operating over lossy ordered channels) by using parameters for the maximum sequence number (**MaxSeqNo**) and the maximum number of retransmissions (**MaxRetrans**). This parameterisation produces an infinite number of instantiations of our CPN model and thus an infinite set of associated occurrence graphs (OGs) that must be analysed. To overcome this problem, we would like to represent this infinite set of occurrence graphs by a symbolic (parametric) OG involving the model parameters.

In CSEC Technical Report 23 [39], such a symbolic OG using algebraic expressions in terms of **MaxSeqNo** and **MaxRetrans** was presented and proved correct. Properties, such as the size of the symbolic OG, absence of unexpected deadlock, absence of livelock, absence of unexpected dead transitions, and channel bounds were proved from the symbolic OG. [39] significantly extended the work documented in CSEC Technical Report 21 [36] which presented algebraic expressions in terms of **MaxSeqNo** only (**MaxRetrans**=0). CSEC Technical Report 21 presented language analysis of the Stop-and-Wait Protocol in terms of **MaxSeqNo** only, a process that was relatively simple due to the simple structure of the symbolic OG (when **MaxRetrans**=0), but this was not lifted to both the **MaxSeqNo** and **MaxRetrans** parameters in CSEC Technical Report 23.

This technical report presents a further step in the parametric verification of the Stop-and-Wait Protocol, that of *Language Analysis*, performed on the parametric SWP in both the **MaxSeqNo** and **MaxRetrans** parameters. We perform language analysis by verifying that the infinite class of Stop-and-Wait Protocols conforms to its service of alternating Send and Receive events. This is done by mapping from the symbolic OG to a *parametric Finite State Automaton* (FSA), an automaton defined in terms of the parameters. Conventional automata reduction techniques were then applied to the parametric FSA. The process of epsilon removal, determinisation and minimisation reduced the parametric FSA to a simple, non-parametric FSA representing exactly the desired service language of alternating send and receive events. Because this language analysis is conducted on the symbolic OG, the verification result holds for all allowable values of the parameters.

Contents

1	Introduction	1
2	Automata and Languages	3
2.1	Languages	3
2.2	Formal Definitions of Finite State Automata	4
2.3	The Language of a Deterministic FSA	5
2.4	Automata Reduction Techniques	6
2.4.1	Epsilon Closure	6
2.4.2	Determinisation using Subset Construction	6
2.4.3	A Discussion of Aspects of Subset Construction for Determinisation	7
2.4.4	Subset Construction using Lazy Evaluation	8
2.4.5	Minimisation	8
2.5	FSA Tool Support	10
3	Previous Work	11
3.1	Non-Symbolic Verification of SWP	11
3.2	Symbolic Analysis using TReX	13
3.3	A Compositional Behavioural Fixed Point Approach	13
3.4	The Sliding Window Protocol	14
3.5	Algebraic Expressions for Other Systems	14
3.6	Our own Previous Work	15
4	The Stop-and-Wait Protocol CPN Model	17
5	Parametric Reachability Graph	19
5.1	Notational Conventions	19
5.2	Marking and Arc Notation	20
5.2.1	Marking Notation	21
5.2.2	Arc Notation	24
5.3	Classifying Markings	27
5.4	Parametric Marking and Arc Notation	28
5.5	Downward-Closed Sets of Markings	28
5.6	The Parametric Reachability Graph	29
5.6.1	Sets of Markings	29
5.6.2	Sets of Arcs	30
5.6.3	The Parametric SWP Reachability Graph	30

6	Parametric Language Analysis	35
6.1	The Stop-and-Wait Service Language	35
6.2	Obtaining a Parametric Representation of the Protocol Language	36
6.3	Epsilon Closures	37
6.3.1	The ϵ -closure of Nodes in $V_{(2a,i)}^{(MS,MR)}$	37
6.3.2	The ϵ -closure of Nodes in $V_{(3b,i)}^{(MS,MR)}$	45
6.4	Determinisation	52
6.5	Minimisation and Conformance to the SWP Service Language	64
7	Conclusions and Future Work	67
	References	68

List of Figures

2.1	A Simple Nondeterministic FSA.	4
2.2	The language equivalent Deterministic FSA of the simple Nondeterministic FSA in Fig. 2.1.	8
2.3	Refining the partition of states based on distinguishability.	9
2.4	The language equivalent Minimised FSA of the Deterministic FSA in Fig. 2.2.	10
4.1	A CPN of the Stop-and-Wait Protocol operating over an in-order medium.	18
4.2	Declarations of the CPN shown in Fig. 4.1.	18
6.1	An FSA of the Stop-and-Wait Service Language.	36
6.2	The markings, represented by mo , ao and mn , reachable from $M_{(2a,i)}^{(MS,MR),(mo,ao,mn,0,ret)}$ through repeated successive firings of <code>receive_mess</code> followed by <code>send_ack</code>	39
6.3	The markings from $V_{(2a,i)}^{(MS,MR)}$ and $V_{(3a,i)}^{(MS,MR)}$, represented by mo , ao and mn , reachable from the markings in Fig. 6.2 through repeated firing of <code>timeout_retrans</code>	41
6.4	The coverage of V_a'' and V_b'' over $V_i^{(MS,MR)}$	43
6.5	The coverage of V_d' , V_e' , V_f' and V_g' over $V_i^{(MS,MR)} \cup V_{i \oplus MS 1}^{(MS,MR)}$	48
6.6	Construction of a deterministic FSA using subset construction with lazy evaluation, showing the initial state, s_0^{det} , and its only successor, C_1	53
6.7	Construction of a deterministic FSA showing the addition of C_2 , the successor of C_1 from Fig. 6.6.	55
6.8	Construction of a deterministic FSA showing the addition of $C_{(3,1)}$, the successor of C_2 from Fig. 6.7.	58
6.9	Construction of a deterministic FSA showing the addition of $C_{(4,1)}$, the successor of $C_{(3,1)}$ from Fig. 6.8.	60
6.10	Construction of a deterministic FSA showing the addition of $C_{(5,1)}$, the successor of $C_{(4,1)}$ from Fig. 6.9.	63
6.11	An abstract visualisation of the parametric deterministic FSA, $DFSA_{RG(MS,MR)}$	65
6.12	An example of the deterministic FSA for <code>MaxSeqNo=2</code> . The <code>MaxRetrans</code> parameter has no effect on the deterministic FSA.	65

List of Tables

5.1	A mapping from an augmented transition name, a_{tn} , to its corresponding enabled binding element, given a source marking, $M = M_{(s_state, r_state), (ssn, rsn), (mo, ao, mn, an, ret)}^{(MS, MR)}$, that enables $TransMap(a_{tn})$	26
5.2	Classification of markings into Classes of markings based on the state of the sender and receiver.	28
5.3	$V_i^{(MS, MR)} = \bigcup_{class \in Class} V_{(class, i)}^{(MS, MR)}$, for $0 \leq i \leq MS$ and $Class = \{1, 2a, 2b, 3a, 3b, 4\}$	30
5.4	The set of arcs, $A_{(1, i)}^{(MS, MR)}$, with source markings in $V_{(1, i)}^{(MS, MR)}$	31
5.5	The set of arcs, $A_{(2a, i)}^{(MS, MR)}$, with source markings in $V_{(2a, i)}^{(MS, MR)}$	31
5.6	The set of arcs, $A_{(2b, i)}^{(MS, MR)}$, with source markings in $V_{(2b, i)}^{(MS, MR)}$	32
5.7	The set of arcs, $A_{(3a, i)}^{(MS, MR)}$, with source markings in $V_{(3a, i)}^{(MS, MR)}$, for $MR > 0$	32
5.8	The set of arcs, $A_{(3b, i)}^{(MS, MR)}$, with source markings in $V_{(3b, i)}^{(MS, MR)}$	33
5.9	The set of arcs, $A_{(4, i)}^{(MS, MR)}$, with source markings in $V_{(4, i)}^{(MS, MR)}$, for $MR > 0$	33
6.1	$DFSA_{RG(MS, MR)}$, where rows 4 and 5 are evaluated for $0 \leq i \leq MS$	64
6.2	The minimised deterministic FSA, $MFSAR_{G(MS, MR)}$ representing the protocol language of $CPN_{(MS, MR)}$	66

Acronyms

ABP	Alternating Bit Protocol
ABRACAD	Alternating Bit sequence numbers, Retransmissions on timeout, Acknowledgements, Connection And Disconnection
BRP	Bounded Retransmission Protocol
CES	Capability Exchange Signalling
CFFD	Chaos-Free Failures Divergences
CPN	Coloured Petri Net
DFSA	Deterministic Finite State Automaton
FAST	Fast Acceleration of Symbolic Transition systems
FIFO	First-In-First-Out
FSA	Finite State Automaton
FSM	Finite State Machine
ITU-T	Telecommunication Standardization Sector of the International Telecommunication Union
LOTOS	Language Of Temporal Ordering Specifications
MaxRetrans	Maximum number of Retransmissions
MaxSeqNo	Maximum Sequence Number
MFSA	Minimised Finite State Automaton
NDFSA	Non-Deterministic Finite State Automaton
PhD	Doctor of Philosophy
PROTEAN	Protocol Emulation and Analysis
RG	Reachability Graph
SDL	Specification and Description Language
SWP	Stop-and-Wait Protocol
TCP	Transmission Control Protocol
TRL	Timed Rewriting Logic
TReX	Tool for Reachability analysis of complex systems

Chapter 1

Introduction

In [36, 37] we began investigations into parametric verification of the class of Stop-and-Wait protocols. A Coloured Petri net (CPN) model of the Stop-and-Wait protocol (SWP) operating over a lossy ordered channel was introduced, parameterised by the maximum sequence number (**MaxSeqNo**) and the maximum number of retransmissions (**MaxRetrans**). These parameters are unbounded, resulting in an infinite set of CPNs and thus an infinite set of associated reachability graphs that need to be analysed. For the initial case of no retransmissions (**MaxRetrans=0**) we developed algebraic expressions over the **MaxSeqNo** parameter, representing the infinite set of reachability graphs over **MaxSeqNo**, **MaxRetrans=0**. These expressions and their proof of correctness can be found in [36, 37].

By explicitly representing the infinite set of reachability graphs for the above class of protocols, we were able to obtain algebraic expressions for a parametric finite state automaton (FSA) representing the protocol language of the SWP CPN for all values of **MaxSeqNo** where **MaxSeqNo** is a parameter. We were able to apply automata reduction techniques directly to this symbolic FSA representation to obtain a single, simple FSA representing the protocol language for all values of **MaxSeqNo** and were thus able to verify that the SWP conforms to its service of alternating send and receive events for all values of **MaxSeqNo** when **MaxRetrans = 0**. The symbolic FSA representation and language analysis performed can also be found in [36, 37].

In [38, 39], we extended this work by generalising the algebraic expressions for the reachability graph of the SWP CPN over *both* parameters, **MaxRetrans** and **MaxSeqNo**. Inclusion of the **MaxRetrans** parameter represented a substantial increase in the complexity of the algebraic expressions. This can be gauged by the size of the RG, which grows linearly in **MaxSeqNo** but quartically in **MaxRetrans** [38, 39] This provided a much more complete verification of the class of SWPs operating over lossy ordered channels, as well as providing more practical significance, as the use of a SWP over a lossy medium without retransmissions is somewhat impractical in the real world.

However, [39] did not perform language analysis. This Technical Report completes the generalisation of the research from [36, 37] to both parameters by performing this language analysis. The procedure followed is similar to that conducted in [36, 37] although complicated by the substantial increase in the complexity of the algebraic expressions that form the parametric reachability graph. A mapping is performed on these expressions to obtain a parametric FSA, to which FSA reduction techniques have been applied. Unlike the relatively simple structure of the parametric FSA from [36, 37] to which simple ϵ -removal and determinisation algorithms [8] can be applied, the more complex structure of this parametric FSA has led us to apply more advanced FSA reduction techniques for ϵ -removal and determinisation, namely ϵ -closure combined with determinisation using subset construction [41, 69] with lazy evaluation of subsets. Following a minimisation procedure, we obtain a single, simple FSA representing the Stop-and-Wait Protocol language for all values of **MaxSeqNo** and **MaxRetrans**, and verify its conformance to the Stop-and-Wait service of alternating Send and Receive events. The material contained in [36, 39] and this report form the basis of [35].

This report assumes that the reader is familiar with the research reported in [36, 39], and assumes general knowledge of the Alternating Bit and Stop-and-Wait Protocols, Coloured Petri Nets and their analysis techniques, and the Protocol Verification Methodology. For an informal introduction to the Alternating Bit and

Stop-and-Wait Protocols, the reader is referred to [36, 39] and the references cited therein. For an introduction to CPNs, the reader is referred to [46, 47, 52]. For details of the protocol verification methodology, the reader is referred to [11, 15, 21].

The rest of this report is organised as follows. Chapter 2 introduces the automata and language concepts used in this report. Chapter 3 discusses previous work in the verification of the Alternating Bit and Stop-and-Wait Protocols, including both the work of others and our own previous work. Chapter 4 presents briefly our CPN model of the Stop-and-Wait Protocol used in [35, 39]. Chapter 5 defines both the notation used for representing the parametric reachability graph, and the algebraic expressions for the parametric reachability graph itself. Chapter 6 contains the major contribution of this report. It performs language analysis by mapping the parametric reachability graph to a parametric Finite State Automaton, and applying automata reduction techniques directly on the algebraic expressions to obtain a minimised, deterministic FSA of the protocol language for the entire class of Stop-and-Wait Protocols. Finally, Chapter 7 presents some concluding remarks and identifies areas of future work.

Chapter 2

Automata and Languages

Formal methods [28] encompass mathematically-based languages, techniques and tools for specifying and analysing systems. Formal methods have a precise, mathematically defined semantics that facilitates rigorous, mathematically based analysis techniques and forms a solid basis for the implementation of computer support tools, indispensable when analysing large systems (e.g. see [40]).

In this report, we make use of two formal techniques, Coloured Petri nets and Finite State Automata. This report assumes knowledge of CPNs, as this is the main modelling tool used by the Computer Systems Engineering Centre. However, to aid understanding of the major contribution of this report (the language analysis in Chapter 6) and provide a measure of self containment, this chapter introduces Finite State Automata along with related topics including languages and automata reduction techniques. The material in this chapter is taken from [35] with additional discussion included (Section 2.4.3).

Finite State Automata (FSA) are finite graphical structures that can be used to represent digital systems that move in discrete steps (from one state to another) [8]. Each node in a FSA is a state of the system and each edge represents a change from one state to another. FSA are particularly useful for recognising or generating a class of languages called *regular languages* [8,41].

2.1 Languages

The relationship between languages and automata is a close one. We begin by defining the following language concepts and notation, based on [8,41,56].

Definition 1 (Alphabet).

An alphabet is a finite, non-empty set of symbols, conventionally denoted Σ .

For example, an alphabet could be the set of all lower case letters, where each such letter is a symbol of the alphabet.

Definition 2 (Strings).

A string (or a word) over an alphabet is a finite sequence of symbols chosen from the alphabet. The set of all non-empty strings over an alphabet, Σ , is denoted Σ^+ . The length of a string is the length of the sequence, i.e. the number of instances of symbols in the string. The string with a length of zero is known as the empty string, denoted ϵ , and may be chosen from any alphabet. The set of all strings, including the empty string, over Σ is denoted Σ^ , i.e. $\Sigma^+ \cup \{\epsilon\} = \Sigma^*$.*

For example, the string “abcd” is a string of length 4, over the alphabet of all lower case letters.

Definition 3 (Language).

A language, \mathcal{L} , is a set of strings over an alphabet, Σ , i.e. $\mathcal{L} \subseteq \Sigma^$. The language containing no strings is known as the empty language, denoted \emptyset , and is a language over any alphabet.*

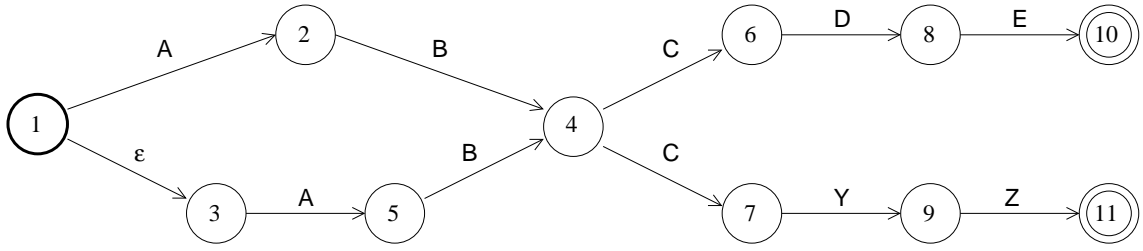


Figure 2.1: A Simple Nondeterministic FSA.

For example, $\mathcal{L} = \{abcd, efgh\}$ is a language over the alphabet of all lower case letters.

Two languages defined over the same alphabet can be compared. This gives rise to the notion of *language inclusion* and *language equivalence*, which can be defined using the set-theoretic concept of subset (based on [50]).

Definition 4 (Language Inclusion).

Given two languages, \mathcal{L}_1 and \mathcal{L}_2 , defined over the same alphabet, we say that \mathcal{L}_1 is included in \mathcal{L}_2 if and only if all strings in \mathcal{L}_1 are also in \mathcal{L}_2 , i.e. $\mathcal{L}_1 \subseteq \mathcal{L}_2$.

Definition 5 (Language Equivalence).

Two languages, \mathcal{L}_1 and \mathcal{L}_2 , defined over the same alphabet, are equivalent if and only if all strings in \mathcal{L}_1 are also in \mathcal{L}_2 and all strings in \mathcal{L}_2 are also in \mathcal{L}_1 , i.e. $\mathcal{L}_1 = \mathcal{L}_2$ iff $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and $\mathcal{L}_2 \subseteq \mathcal{L}_1$.

2.2 Formal Definitions of Finite State Automata

The following formally defines a Finite State Automaton as used in this report, based on [12, 50, 56].

Definition 6 (Finite State Automaton).

A Finite State Automaton, FSA, is defined as a 5-tuple, $FSA = (S, \Sigma, \Delta, s_0, F)$, where:

- S is the set of states of the FSA;
- Σ is the set of input symbols (the alphabet) of the FSA;
- $\Delta \subseteq S \times (\Sigma \cup \epsilon) \times S$ is the set of edges (the transition relation) of the FSA labelled with an element of the alphabet or ϵ ;
- $s_0 \in S$ is the initial state of the FSA; and
- $F \subseteq S$ is the set of halt states (final states) of the FSA.

This is the most general definition of an FSA used in this report. It covers both nondeterministic and deterministic FSAs, with or without ϵ (empty) moves. Note that we use edges to describe the arcs of a FSA, to avoid confusion with the arcs in a reachability graph.

An FSA has a graphical representation. As an example, consider the nondeterministic FSA in Fig. 2.1. By convention, the states of the FSA are represented by ellipses and the edges are represented by arcs between states, labelled by elements from the alphabet. By convention, the initial state is drawn in bold and halt states are denoted by two concentric ellipses. In Fig. 2.1 the states have been numbered in a breadth-first manner, from 1 to 11. State 1 is an initial state (drawn in bold) and states 10 and 11 are halt states (drawn as double circles). The alphabet of this FSA is the set of symbols $\{A,B,C,D,E,Y,Z\}$.

Finite State Automata operate through a sequence of moves. Each state has a number of actions (outgoing edges), labelled by symbols from the alphabet or ϵ , corresponding to allowable moves from the current state

to the *next* state. Starting from the initial state, s_0 , each move is dictated by the current state and the set of actions that may occur from that state. (As a language recogniser, consider the next state to be determined by the value of an input symbol read from a string. Edges labelled with ϵ can be traversed without consuming an input symbol.) In Fig. 2.1, note the nondeterministic choice introduced by the ϵ move from state 1 to state 3, and the nondeterministic choice on symbol C from state 4.

An *accessible* state is a state that is *reachable* from the initial state via some sequence of actions. This can be defined in a similar way to a reachable marking of a CPN [8,58]:

Definition 7 (Accessible state).

For a FSA, $FSA = (S, \Sigma, \Delta, s_0, F)$, a state, $s_n \in S$, $n > 0$, is *accessible* (from s_0) iff there exists states $s_1, s_2, \dots, s_{n-1} \in S$ such that for all $i \in \{0, \dots, n-1\}$, $(s_i, l_i, s_{i+1}) \in \Delta$.

For example, all states in the FSA in Fig. 2.1 are accessible.

A Deterministic FSA (DFSA) is a special case of Definition 6 where there is at most one transition on each input symbol for each state, and there are no ϵ moves, i.e. each successor state is uniquely determined by the current state and the next input symbol. Formally (based on [58]):

Definition 8 (Deterministic Finite State Automaton).

A FSA, $FSA = (S, \Sigma, \Delta, s_0, F)$, is a DFSA, if its transition relation, Δ , is a many-to-one or one-to-one relation, i.e. $\forall (s, l, s') \in \Delta, l \neq \epsilon$, and $\forall s \in S$, if $(s, l, s') \in \Delta$ and $(s, l, s'') \in \Delta$, then $s' = s''$.

Hence, for a DFSA, we can define a *next state* function [41,50,56,57,69], $\delta : S \times \Sigma \rightarrow S$, where $\delta(s, l) = s'$ if and only if $(s, l, s') \in \Delta$, which returns the unique next state for a given state and input symbol.

2.3 The Language of a Deterministic FSA

The *language* of an FSA is the set of strings over the alphabet of the FSA that are *accepted* by the FSA, i.e. strings, σ , that begin in the initial state, s_0 , traverse the FSA through a sequence of allowable moves, and end in a final state, $s \in F$. Hence, we see that the language accepted by the FSA in Fig. 2.1 contains two strings, ABCDE and ABCYZ. The formal definition of the language of a nondeterministic FSA is not needed in this report, but can be found in [41]. The language of a deterministic FSA can be defined formally by firstly defining an *extended transition* function, inspired by [41, 58].

Definition 9 (Extended Transition Function).

Let DFSA = $(S, \Sigma, \delta, s_0, F)$ be a deterministic FSA. An *extended transition function*, $\hat{\delta} : S \times \Sigma^+ \rightarrow S$, which takes strings as input, is defined as

$$\hat{\delta}(s, l\sigma) = \begin{cases} \delta(s, l), & \text{if } \sigma = \epsilon \\ \hat{\delta}(\delta(s, l), \sigma), & \text{if } \sigma \in \Sigma^+ \end{cases}$$

where $s \in S$ and $l \in \Sigma$, and juxtaposition is used for string concatenation.

Unlike [41], we do not include a mapping, $\hat{\delta}(s, \epsilon) = s$, in our definition of $\hat{\delta}$. The intention is for this to represent the change in state upon receiving no input, but the overloaded use of ϵ for this purpose may be confusing (recall that we are dealing with a DFSA, with no ϵ moves).

Definition 10 (Language of a DFSA).

For a deterministic FSA, DFSA = $(S, \Sigma, \delta, s_0, F)$, the *language accepted by DFSA*, denoted $\mathcal{L}(DFSA)$, is given by

$$\mathcal{L}(DFSA) = \begin{cases} \{\sigma \in \Sigma^+ \mid \hat{\delta}(s_0, \sigma) \in F\}, & \text{if } s_0 \notin F \\ \{\sigma \in \Sigma^+ \mid \hat{\delta}(s_0, \sigma) \in F\} \cup \{\epsilon\}, & \text{if } s_0 \in F \end{cases}$$

Unlike [58], we include the possibility of a DFSA accepting the empty string when the initial state is also a halt state. (In [41] this is implicit in the extended transition function.)

2.4 Automata Reduction Techniques

Every non-deterministic FSA can be represented by a language equivalent deterministic FSA [8]. This process involves two steps: removal of all empty moves (first empty cycles, then the remaining empty moves), and transformation into a deterministic FSA using a subset construction technique [8].

2.4.1 Epsilon Closure

A more recent technique for epsilon removal, one that does not require the two step process of removal of empty cycles then removal of remaining empty moves, is that of *epsilon closure* [41,69]. ϵ closure involves computing the transitive closure of the (not necessarily connected) subgraph of the FSA consisting of all ϵ moves. It is, essentially, the algorithm for epsilon removal as presented in [8] but formulated to apply to the whole FSA. A variant of this procedure is used in AT&T's Finite State Machine (FSM) Library [34], which may result in the inclusion of inaccessible states (states not reachable from the initial state) which require deletion. For more details, see [63,84].

The ϵ -closure operation defines all states in an FSA reachable from a given state via a sequence of 0 or more ϵ moves. We define the ϵ -closure operation based on the narrative description in [41].

Definition 11 (ϵ -closure).

For FSA = $(S, \Sigma, \Delta, s_0, F)$ the ϵ -closure of $s \in S$ is given by $Closure : S \rightarrow 2^S$, where $Closure(s) = \{s' \mid s \xrightarrow{\epsilon^*} s', s' \in S\}$ and $s \xrightarrow{\epsilon^*} s'$ indicates that s' can be reached from s via a contiguous sequence of 0 or more ϵ moves.

Note that $s \in Closure(s)$. This function can be extended to run over sets of markings:

Definition 12 (Linear extension of ϵ -closure).

For FSA = $(S, \Sigma, \Delta, s_0, F)$ the ϵ -closure of $S' \subseteq S$ is given by $CLOSURE : 2^S \rightarrow 2^S$, where $CLOSURE(S') = \bigcup_{s \in S'} Closure(s)$.

It is useful to define a function that returns all non- ϵ outgoing edges from a set of states in a FSA:

Definition 13 (non- ϵ outgoing edges from a set of states).

For FSA = $(S, \Sigma, \Delta, s_0, F)$ a function that maps from a set of states, $S' \subseteq S$, to the set of non- ϵ outgoing edges of the states in S' , is given by $OutEdges : 2^S \rightarrow 2^\Delta$ where $OutEdges(S') = \{(s', l, s'') \mid s' \in S', (s', l, s'') \in \Delta, l \neq \epsilon\}$

If S' is an ϵ -closure (as will be our intention) then this function returns all non- ϵ outgoing edges of the ϵ -closure.

2.4.2 Determinisation using Subset Construction

A standard subset construction technique is presented in [8, 41, 69] for determinisation of an ϵ -free FSA. Essentially, the subset construction technique builds a deterministic FSA whose states are the set of subsets of the states in the non-deterministic FSA. The subset construction technique is extended in [41, 69] by combining subset construction with ϵ -closure to avoid the need for explicit removal of empty cycles and empty moves prior to determinisation.

Formalising the narrative procedure presented in [8] and introducing ϵ -closure as in [41, 69], given a non-deterministic FSA, $NDFSA = (S, \Sigma, \Delta, s_0, F)$, an equivalent deterministic FSA, $DFSA = (S^{det}, \Sigma, \Delta^{det}, s_0^{det}, F^{det})$, is produced using the subset construction technique as follows:

1. The set of states of $DFSA$ is the set of subsets of the states of $NDFSA$ excluding the empty set, i.e. $S^{det} = 2^S \setminus \emptyset$. This may be a large set. Often, in practice, many of the states in S^{det} will not be accessible from the start state of $DFSA$, and so can be effectively *thrown away* [41], i.e. $S^{det} \subseteq 2^S \setminus \emptyset$.

2. Consider state $s^{det} = \{s_1, s_2, \dots, s_n\}$, $s^{det} \in S^{det}$ and $s_1, s_2, \dots, s_n \in S$. For each $l \in \Sigma$, let $s^{det'} = \{s'' \mid (s, l, s') \in \Delta, s'' \in Closure(s'), s \in s^{det}\}$. If $s^{det'} \neq \emptyset$, add to Δ^{det} the arc $(s^{det}, l, s^{det'})$. Repeat for all $s^{det} \in S^{det}$.
3. The initial state of *DFSA* is the ϵ -closure of the initial state of *NDFSA*, i.e. $s_0^{det} = Closure(s_0)$.
4. The final states of *DFSA* are all states in S^{det} which contain a final state of *NDFSA*, i.e. for all $s^{det} \in S^{det}$, $s^{det} \in F^{det}$ if and only if $\exists s \in s^{det}$ such that $s \in F$.

Formally:

Definition 14 (Determinisation).

Let *NDFSA* = $(S, \Sigma, \Delta, s_0, F)$ be a non-deterministic FSA. A language equivalent deterministic FSA is given by *DFSA* = $(S^{det}, \Sigma, \Delta^{det}, s_0^{det}, F^{det})$, where:

- $S^{det} \subseteq 2^S \setminus \emptyset$;
- $\Delta^{det} = \{(s^{det}, l, s^{det'}) \mid l \in \Sigma, s^{det} \in S^{det}, s^{det'} = \cup_{s \in s^{det}} \{s'' \mid (s, l, s') \in \Delta, s'' \in Closure(s')\}, s^{det'} \in S^{det}, s^{det'} \neq \emptyset\}$;
- $s_0^{det} = Closure(s_0)$; and
- $s^{det} \in F^{det} \iff \exists s \in s^{det} \mid s \in F$.

The two differences between the subset construction technique combining ϵ -closure and the original subset construction technique are [41]:

1. The initial state of *DFSA* is the ϵ -closure of the initial state of *NDFSA*, instead of simply the singleton set containing the initial state of *NDFSA*; and
2. The successor of a state $s^{det} \in S^{det}$ on action $l \in \Sigma$ is the ϵ -closure of the successors of all $s \in s^{det}$ on input symbol l , instead of simply the successors of all $s \in s^{det}$ on input symbol l .

2.4.3 A Discussion of Aspects of Subset Construction for Determinisation

Both the inclusion and exclusion of the empty set, \emptyset , from S^{det} has support in the literature, arising naturally from the definition of the transition relation. Unlike our formalism, [8, 41, 69] all express the transition relation of a nondeterministic FSA as a (partial) function from a state and action to a set of states.

In [8] the determinisation using subset construction procedure is given as an algorithm which does not consider \emptyset to be a member of S^{det} . The transition relation of an FSA is given as a partial function, i.e. not all actions are defined from all states. The issues arising from a partial transition relation in subset construction are not mentioned or discussed.

The procedure given in [69] does consider $\emptyset \in S^{det}$. The transition function of the nondeterministic FSA is implicitly defined in [69] as a total function by mapping each undefined state and action pair to \emptyset . When the transition function maps a state and an action to \emptyset the state consisting of \emptyset appears naturally in the equivalent deterministic FSA, when performing subset construction. The transition relation of the resulting deterministic FSA then implicitly maps the state \emptyset to itself on the occurrence of every action, although this is not stated explicitly in [69].

In [41] the transition function is again defined as a total function, mapping to \emptyset any state and action pair not defined in the nondeterministic FSA. Through the same process as above, the state \emptyset can appear in the equivalent deterministic FSA and again, implicitly, the transition function of the deterministic FSA will map the state \emptyset to itself on all actions. However, [41] does discuss this empty state. It points out that an FSA that does not define a successor for every state on every action is, technically, not a deterministic FSA. Converting it to a deterministic FSA in the strict sense introduces a so-called *dead state*, which can be labelled by \emptyset , allowing a single action, leading to a single successor, to be defined for every state on every input.

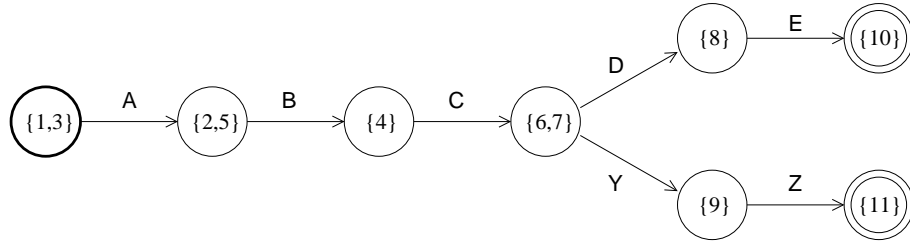


Figure 2.2: The language equivalent Deterministic FSA of the simple Nondeterministic FSA in Fig. 2.1.

Although not explicitly discussed in [41, 69], the empty state introduced by subset construction can be safely ignored. The intuitive reasoning is that this state is a non-accepting state, which leads to itself on every input symbol. It therefore contributes nothing to the language accepted by the FSA (or conversely the language generated by the FSA). Therefore, in this report, we choose to ignore the empty state \emptyset in our subset construction, but with sound reasoning.

2.4.4 Subset Construction using Lazy Evaluation

A lazy approach to subset construction is also presented in [41]. The general idea is to generate the deterministic FSA starting from the initial state of the nondeterministic FSA, evaluating subsets of states only as needed. This method therefore only generates the subsets of states (and the corresponding edges) that are actually accessible from the initial state of the deterministic FSA and thus there is no need to eliminate inaccessible states.

Inspired by [41], an algorithm for producing *DFSA* from *NDFSA* using subset construction with lazy evaluation can be defined as follows.

1. The initial marking of *DFSA*, $s_0^{det} = \text{Closure}(s_0)$, is guaranteed to be accessible, i.e. $s_0^{det} \in S^{det}$.
2. Select a state, $s^{det} = \{s_1, s_2, \dots, s_n\}$, that has been found to be accessible, i.e. $s^{det} \in S^{det}$. For each input symbol, $l \in \Sigma$, compute the set of states, $s^{det'} = \text{CLOSURE}(\{s' \mid (s, l, s') \in \Delta\})$. The state $s^{det'}$ is accessible, i.e. $s^{det'} \in S^{det}$, and $(s^{det}, l, s^{det'}) \in \Delta^{det}$.
3. Repeat step 2 until no more new accessible states of *DFSA* are discovered.

Figure 2.2 shows the language equivalent deterministic FSA resulting from applying this algorithm to the FSA in Fig. 2.1. This lazy evaluation method is used symbolically for determinisation in Chapter 6.

2.4.5 Minimisation

Every deterministic FSA can be represented by a language equivalent deterministic FSA with a minimum number of states. This minimum FSA is unique, up to a relabelling of states (isomorphism) [41]. The process of obtaining a minimal representation is known as *minimisation*.

Both [41] and [8] define a process of minimisation based on the notion of *equivalence* of states in the FSA. Two states in the FSA are said to be equivalent if and only if they both accept exactly the same set of strings over the alphabet of the FSA. Formally [41, 58]:

Definition 15 (Equivalence of states).

Let $DFSA = (S, \Sigma, \delta, s_0, F)$ be a deterministic FSA. Two states, $s, s' \in S, s \neq s'$, are equivalent, denoted $s \approx s'$, if and only if $\text{Strings}_{DFSA}(s) = \text{Strings}_{DFSA}(s')$, where $\text{Strings}_{DFSA} : S \rightarrow \Sigma^*$ is a function that returns the set of strings accepted by a state of *DFSA*, defined as:

$$\text{Strings}_{DFSA}(s) = \begin{cases} \{\sigma \in \Sigma^+ \mid \hat{\delta}(s, \sigma) \in F\}, & \text{if } s \notin F \\ \{\sigma \in \Sigma^+ \mid \hat{\delta}(s, \sigma) \in F\} \cup \{\epsilon\}, & \text{if } s \in F \end{cases}$$

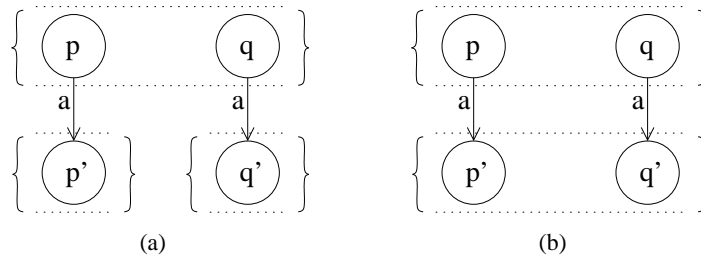


Figure 2.3: Refining the partition of states based on distinguishability.

s and s' are not equivalent, denoted $s \not\approx s'$, if and only if $Strings_{DFSA}(s) \neq Strings_{DFSA}(s')$.

Thus, a minimised FSA (MFSA) can be formally defined as a DFSA in which there are no equivalent states (based on [41, 58]).

Definition 16 (Minimum Finite State Automaton).

A deterministic FSA, $DFSA = (S, \Sigma, \delta, s_0, F)$, is a minimum FSA if and only if no two states are equivalent, i.e. $DFSA$ is a single state FSA; or $\forall s \in S \setminus \{s_0\}$, s is accessible from s_0 , and $\forall s, s' \in S$, ($s \neq s'$), $s \not\approx s'$.

Although both [41] and [8] present different algorithms for generating the minimum FSA from a deterministic FSA, they have a key underlying element in common: identifying *distinguishable* states.

Two states are distinguishable if they are not equivalent [41], i.e. if they each accept a different set of strings. To obtain the minimised FSA, a partition is defined on the set of states and refined so that it divides the states into disjoint subsets, where each state in a subset is equivalent to every other state in the same subset, and distinguishable from all other states in all other subsets.

The partition is initially defined to divide the states into two subsets, based on halt state status. Halt states and non-halt states are immediately distinguishable as halt states can accept the empty string but non-halt states cannot. Thus, immediately, all halt states are distinguishable from all non-halt states, and vice versa. The partition is then subsequently refined using the following procedure.

Two states, p and q , from the same subset are distinguishable if they either accept different sets of input symbols, or can be distinguished by some single input symbol in the following way. Suppose p and q both accept the same input symbol a , leading to p' and q' respectively. If p' and q' are distinguishable, then p and q are distinguishable. This process continues until no more pairs of distinguishable states can be discovered, i.e. no more subdivisions are possible. As an example, consider states p, q, p' and q' as portrayed in a fragment of an FSA in Fig. 2.3. In Fig. 2.3 (a), the states p' and q' are in different subsets, indicated by the braces and dotted lines. Hence, states p and q are distinguishable on action a , and the partitioning of states should be refined to separate p and q into different subsets. In Fig. 2.3 (b), however, p' and q' are in the same subset, and thus p and q are not distinguishable on action a .

The resulting subsets of states are used to formulate the minimised FSA. The arcs of the minimised FSA arise naturally from the subsets of states, as every state in a subset will accept the same input symbols, and on acceptance of an input symbol will lead to successor states that all belong to the same subset. Hence, the subsets of states are treated as single compound states. The initial state of the minimised FSA is the subset of states containing the initial state of the deterministic FSA. The halt states of the minimised FSA are those subsets of states composed of halt states of the deterministic FSA. (Note that either all states in a subset are halt states, or none are.)

Formalising this narrative description from [8], the minimised FSA, $MFSA$, of a deterministic FSA, $DFSA$, is defined as follows.

Definition 17 (Minimised FSA of a Deterministic FSA).

Let $DFSA = (S^{det}, \Sigma, \Delta^{det}, s_0^{det}, F^{det})$ be a deterministic FSA. A language equivalent minimum FSA is given by $MFSA = (S^{min}, \Sigma, \Delta^{min}, s_0^{min}, F^{min})$, where:

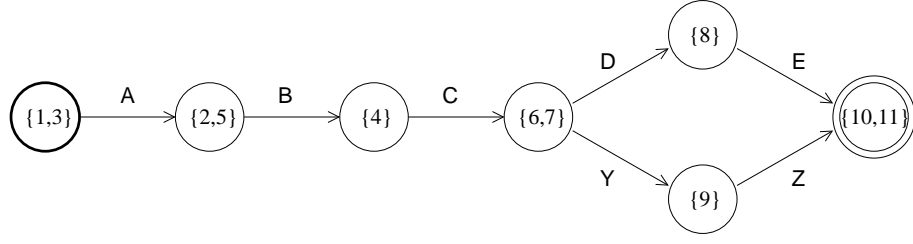


Figure 2.4: The language equivalent Minimised FSA of the Deterministic FSA in Fig. 2.2.

- $S^{min} \subseteq 2^{S^{det}} \setminus \emptyset$ is a partition of the states in S^{det} , (i.e. $\forall s^{min}, s^{min'} \in S^{min}, s^{min} \neq s^{min'} \implies s^{min} \cap s^{min'} = \emptyset$), such that $s^{min} \in S^{min}$ if and only if $\forall s_1^{det}, s_2^{det} \in s^{min}$:
 1. $s_1^{det} \in F^{det} \iff s_2^{det} \in F^{det}$; and
 2. $\forall l \in \Sigma, (s_1^{det}, l, s_1^{det'}) \in \Delta^{det} \iff (s_2^{det}, l, s_2^{det'}) \in \Delta^{det}$ and $\exists s^{min'} \in S^{min}$ such that $s_1^{det'}$ and $s_2^{det'}$ $\in s^{min'}$.
- $\Delta^{min} = \{(s^{min}, l, s^{min'}) \mid s^{min}, s^{min'} \in S^{min} \text{ and } (s^{det}, l, s^{det'}) \in \Delta^{det} \text{ for some } s^{det} \in s^{min}, s^{det'} \in s^{min'}\}$;
- $s_0^{min} \in S^{min} \mid s_0^{det} \in s_0^{min}$; and
- $F^{min} \subseteq S^{min} \mid s^{min} \in F^{min} \implies s^{min} \subseteq F^{det}$

Figure 2.4 shows the language equivalent minimised FSA resulting from applying the minimisation algorithm to the deterministic FSA in Fig. 2.2.

2.5 FSA Tool Support

Manipulation of FSAs, including epsilon removal and determinisation (as separate steps) and minimisation operations, is ably supported by AT&T's FSM Library [34]. However, tools from the FSM Library were only used in support of the work in this report and not used directly.

Chapter 3

Previous Work

This chapter presents an extended and updated literature review from that in [39]. This is based on the corresponding chapter in [35].

When discussing previous work in the area of parametric verification, the precise meaning of ‘parametric’ must be made clear. There is a distinction between parametric systems where the parameter is the number of identical instances of an entity within a system, as in e.g. [10, 23, 45, 54, 55, 62, 67, 74] and [65, 86] for surveys of techniques, and a parameter that influences the operation of an entity or a fixed number of entities. In the context of our work, ‘parametric’ refers to systems of the latter type. Accordingly, this chapter limits the discussion of parametric verification literature to work done previously in addressing the verification of the latter type of parametric system.

This chapter is organised as follows. Section 3.1 introduces relevant literature on the verification of the Stop-and-Wait and Alternating Bit protocols in a non-parametric setting. Sections 3.2 and 3.3 describe and discuss two techniques applied to the parametric verification of the Stop-and-Wait Protocol. The approach described in Section 3.3 has been applied to the Sliding Window Protocol, which we discuss in Section 3.4. Previous work on developing algebraic expressions for parametric verification of other systems is discussed in Section 3.5. Finally, our own previous work is discussed in Section 3.6.

3.1 Non-Symbolic Verification of SWP

The simplest Stop-and-Wait Protocol (SWP) [70, 77] restricts its sequence numbers to 0 and 1 and is known as the Alternating Bit protocol (ABP) [9]. The ABP features extensively in the literature [4, 5, 9, 20, 30, 66, 72, 75–77, 79]. Many demonstrate that the ABP will work perfectly over an underlying in-order medium, a medium that behaves in a FIFO (First-In First-Out) manner and that may also include loss, as discussed below.

The original ABP [9] was designed to work between two terminals (entities) over a half duplex line. Only one message can be in transit in either direction at any one time, hence (in a sense) this protocol operates over a FIFO medium, as no overtaking can occur on such a line. An automaton describing the behaviour of each of the two entities is provided. Given an unbounded number of retransmissions and a lossless (but corrupting) medium, the authors show that the ABP is infallible, provided all transmission errors can be detected. A proof of this follows from the proof of infallibility given in [60] for Lynch’s Protocol, a predecessor to the ABP.

The ABP is often used as a case study when developing a new modelling language or a derivation from an existing modelling language, to demonstrate the use or effectiveness of the new language. This is the case in [72] where the ABP is used as an example to illustrate a new Timed Rewriting Logic (TRL) for capturing the static and dynamic aspects of SDL (Specification and Description Language) [43]. This paper defines a Timed rewriting logic semantics for SDL and uses the ABP to demonstrate this new formal semantics. The ABP variant used is the original ABP from [9] but operating over a medium that may lose messages but cannot corrupt or reorder them (i.e. non-lossy FIFO). A timeout mechanism for retransmissions is discussed, which is not present in [9]. Timing properties relating to the performance of this variant of the ABP are investigated.

The results do not mention a bound on the channels, nor a comparison of the protocol to a service specification.

Another example of the ABP being used as a case study is in [75] and [76] where the ABP is formally modelled and analysed using Temporal Petri nets (derivations of Petri nets with restrictions on the firing of transitions based on formulae containing temporal operators.) This paper presents a model of the ABP operating over channels that allow detectable loss and corruption but not duplication or reordering of data (i.e. lossy FIFO). A timeout mechanism with unbounded retransmissions, not present in the original ABP [9], is discussed and incorporated into the model. Retransmissions occur when the medium signals to the sender that loss has occurred upon the sender sending a message or the receiver sending an acknowledgement. This is a rather significant modelling decision, as in practice a sending entity will not be able to detect loss in the underlying medium, nor will the underlying medium inform the sender that a message or acknowledgement were lost. The ABP is proved correct under the following assumptions: the sender and receiver processes do not halt, and if a message is sent then eventually it is either delivered without corruption, with corruption, or is lost (progress); and if the sender sends messages infinitely often, then the receiver will receive uncorrupted messages infinitely often (fairness). It is proved correct by showing that the Temporal Petri Net possesses certain safety and liveness properties: The sequence of data items that have been output to the receiver user by the receiving entity at a given moment is a prefix of the sequence given to the sending entity by the sender user (safety); and any data item given to the sending entity by the sender user is eventually output to the receiver user by the receiving entity (liveness).

Afek et al [5] also analyse the ABP in terms of safety and liveness properties. They present the ABP in the form of two deterministic finite state automata that communicate via a channel. The paper assumes unbounded retransmission, that the underlying channel is FIFO, and that all messages are either received error free and in the order sent, or are lost (message corruption is treated as message loss.) They present a protocol that is more generic than the original ABP, as they allow more than two sequence numbers in the sequence number space. The reason for this is they present a variant that is *self-stabilising*, i.e. that will converge to normal operation in the event of sender and receiver losing synchronisation. Their model guarantees three things: convergence to stable operation (stabilisation); sequences of data elements written to the output of the receiver during a working interval are always a prefix of the sequence of data elements read from the input of the sender during the same interval (safety); and within a finite time one more data elements will be written to the output of the receiver (liveness).

The Abracadabra Service and Protocol Example [79] describes a protocol that uses Alternating Bit sequence numbers, Retransmissions on timeout, Acknowledgements, and includes Connection And Disconnection procedures (ABRACAD), and is one of a graded set of examples used to provide guidelines for the application of three standardised formal description techniques, namely Estelle [24], LOTOS (Language Of Temporal Ordering Specifications) [22] and SDL [43]. The underlying medium used in this example may lose messages but may not corrupt, reorder, invent or duplicate messages. Retransmissions are bounded by an integer parameter. Models are created using each of these three formal description techniques, and a subjective assessment of each method is given by the author. However, no formal analysis results are presented.

Billington et al [20] use a variant of the ABP (the single frame procedures of the D-Channel Link Level Protocol for Basic Access to the Integrated Services Digital Network contained in draft Recommendation [26]) to demonstrate a software tool called PROTEAN (PROTocol Emulation and ANalysis). A Numerical Petri Net model was created, using a medium that may lose messages but does not corrupt, duplicate or reorder them. Retransmissions are used to recover from frame loss. Channel bounds and an explicit maximum number of retransmissions are parameters of the model. The correctness of the protocol was verified using reachability analysis to investigate deadlocks (no undesired deadlocks were found) and language analysis to determine whether the protocol conformed to its service. Incorrect sequences (sequences of actions not specified in the service language) were discovered in the protocol.

Abdulla et al [4] demonstrate that a number of verification problems (including reachability and safety) are decidable for systems consisting of finite state processes communicating over unbounded lossy FIFO channels. The Alternating Bit Protocol with unbounded retransmissions is one example they use. Labelled transition systems are used to model the ABP and the authors argue the correctness of algorithms to verify the reachability,

safety and eventuality properties defined. A safety property is stated for the ABP saying that no two send or receive actions can be performed consecutively by the sender, essentially defining a service for the ABP and confirming that the ABP conforms to this service.

Reisig [66] develops the ABP in a series of steps as part of a case study on acknowledged messages, developed incrementally using simple Petri net models to illustrate the principles and operation of the ABP over FIFO communication channels. The models assume that messages may get lost (only finitely many consecutive messages may get lost) but are never falsified. Unbounded retransmission of messages, unique (unbounded) message IDs (sequence numbers), and then alternating sequence numbers are progressively introduced. No formal analysis is conducted. The ABP is also used to illustrate the modelling of protocols using Petri Nets in [30], and again, no analysis is performed.

Tanenbaum [77] develops the Stop-and-Wait and Alternating Bit protocols from first principles and then conducts basic analysis on variants of these protocols using finite state machine and Petri net models. Bounded sequence numbers and unbounded retransmission on timeout are features of the protocols that are developed.

None of this literature addresses the issue of parametric verification of the ABP or SWP, for an arbitrary maximum sequence number and an arbitrary (rather than unbounded) maximum number of retransmissions. We now shift our attention to literature with more of a parametric flavour.

3.2 Symbolic Analysis using TReX

The ABP and another variant called the Bounded Retransmission Protocol (BRP) are used in [3] to demonstrate a symbolic verification methodology [1]. TReX (Tool for Reachability Analysis of Complex Systems) [6, 78] was used to implement this methodology in [3]. The content of unbounded lossy FIFO channels is modelled by (a restricted class of) regular expressions thus providing a symbolic representation of the channels. TReX uses an acceleration technique [2] to calculate the effect of firing transitions an arbitrary number of times (sometimes called meta-transitions). This allows a small symbolic state space to be calculated based on the states of the sender and receiver ABP processes. They verify that the ABP conforms to its service of alternating sends and receives, using the Aldebaran tool [25]. In the BRP example, the maximum number of retransmissions was not explicitly modelled as a parameter, but rather as a nondeterministic upper limit, giving rise to a single, infinite-state model, i.e. the sender could retransmit an arbitrary number of times before deciding that the retransmission limit had been reached (hence the model is actually modelling unbounded, but finite, retransmission). This differs from the work in [35, 36, 39] and this report, in which the maximum number of retransmissions is modelled explicitly as a parameter, and thus giving rise to an infinite number of finite-state systems, one for each value of `MaxRetrans`. We also model an arbitrary maximum sequence number as a parameter, rather than being limited to a maximum sequence number of 1.

3.3 A Compositional Behavioural Fixed Point Approach

Valmari and his co-workers (e.g. [80,82,83]) use a behavioural fixed point method and compositional techniques for the verification of parametric systems. In [82] a variant of the ABP using limited retransmission, i.e. where there is an arbitrary bound (e.g. `MaxRetrans`) on the number of retransmissions, is verified using Chaos-Free Failures Divergences (CFFD) equivalence [81, 83]. The behavioural fixed point method allows the modelling of `MaxRetrans` as a symbolic integer parameter. This variant of the ABP is found to behave correctly for an unbounded number of retransmissions provided there is a finite upper bound to the number of consecutive messages and acknowledgements that may be lost in the channel.

The following comparison is taken from [17, 18]. There are several differences with the work presented in [35, 36, 39] and this report. Perhaps the most significant is that the channels are limited to a capacity of one, whereas the channels considered in our work are unbounded. Valmari [82] considers this to be a more difficult problem. Valmari's method relies on defining a separate counter process which needs to be synchronised (using parallel composition) with the sender logic, which has 18 states. The counter itself is a recursive parallel

composition of counter cells. The receiver is a relatively straightforward 6 state process. The acknowledgement channel is given as a 3 state process, but the data channel is more complex and not given explicitly in the paper. To obtain the model, all these processes are synchronised with parallel composition. In contrast, the CPN model presented in Chapter 4 (from [35, 39]) integrates all these aspects in the one model, and extends the model to include unbounded FIFO queues and sequence numbers with an arbitrary maximum sequence number as a parameter. However, our model does not have explicit communication with the users (but relies on the send and (non-duplicate) receive transitions to be considered as synchronised communication with the user) and does not consider reporting errors to the user. There is no technical reason to prevent extension of our model to include these features, however our aim is to verify the protocol itself and thus explicit modelling of communication with the user is unnecessary for our purposes.

3.4 The Sliding Window Protocol

Previous work in a similar vein involves verification of the Sliding Window protocol [73], an extension of the SWP allowing multiple outstanding messages at the sender. The Sliding Window protocol reduces to the SWP when both sender and receiver window sizes are 1. Kaivola [48, 49] examines the Sliding Window protocol for various window sizes, unbounded retransmissions, arbitrary channel capacity and modulo (wrapping) sequence numbers. The compositional technique from [80, 82, 83] is used, in conjunction with abstraction, to replace components of the system with simpler ones, but which still preserve the externally visible behaviour and all the properties to be verified. Each channel is modelled by the composition of a one-place buffer with itself $n - 1$ times, and with a one-place lossy buffer to model loss, to give a channel with total capacity of n , however it is proved that the results are valid for channels of any capacity. Data independence principles [68, 85] are applied and the analysis covers both finite and infinite sequences of input data. The Sliding Window protocol is verified only for small concrete values of both the maximum sequence number parameter and the sending and receiving window sizes. In contrast, our work has considered only the simplest version of the Sliding Window protocol, i.e. the SWP, but it verifies the SWP for every possible concrete value of the maximum sequence number and maximum retransmission parameters (rather than unbounded retransmission), relies on patterns in the reachability graph rather than a compositional approach, and does not consider data explicitly.

A less recent but still relevant result by Knuth [51], referenced by Kaivola [48], also deals with verification of link level protocols in a very general sense, incorporating both the SWP and sliding window protocols. The main thrust of the paper is that for unbounded retransmissions and channels with a limited (but measurable) amount of re-ordering of messages and acknowledgements, it is possible to define a lower bound for the maximum sequence number parameter to guarantee correct operation of the protocol, in the sense that every message sent by the sender will be uniquely and correctly identified by the receiver. In contrast, our work provides a symbolic treatment of the SWP over channels that do not reorder, and as mentioned previously, models the maximum number of retransmissions parameter explicitly. It thus provides verification results for every concrete value of this parameter, rather than unbounded retransmissions which are impractical in the event of, for example, a broken link.

3.5 Algebraic Expressions for Other Systems

There are two notable pieces of work dealing with the development of algebraic expressions to represent the reachability graphs of infinite families of systems. The first is documented in [19, 40]. The Data Transfer service of the Transmission Control Protocol (TCP) has been modelled using a Coloured Petri net, parameterised by the capacity of the medium. The reachability graph of this parametric system grows exponentially in the size of the medium, which in general is unbounded. This results in an infinite family of automata representing TCP's data transfer service language. Closed form algebraic expressions were formulated, in terms of the medium capacity parameter, to represent the family of reachability graphs of the Data Transfer Service CPN. Once the initial state and halt states were designated, the resulting family of automata was proved to be minimum

and deterministic. Thus no further automata reduction was required. The key differences with our work are that these expressions deal with a service specification, not a protocol specification, the Data Transfer service CPN is parameterised in a single parameter only, the parametric reachability graph can be mapped directly to a parametric minimised deterministic FSA without the need for FSA reduction techniques, and that no attempt is made to verify TCP against the service.

The second use of algebraic expressions, in an approach developed in parallel with our work [58, 59], is in the specification and analysis of the service of the Capability Exchange Signalling (CES) protocol, developed by the Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T) as part of ITU-T recommendation H.245 [44], *Control protocol for multimedia communications*. In [58, 59], a model of the CES service is created, parameterised by the capacity of the service. Recursive formulae for the parametric reachability graphs are discovered and proved correct for the CES service for two cases: when the service provides for in-order delivery which may be lossless or lossy. In the lossless case, the recursively defined parametric reachability graph maps directly to a recursively defined finite state automaton that is both minimum and deterministic (as was the case for TCP). Also, closed-form expressions for this parametric reachability graph are presented in [58]. In the lossy case, however, when mapping to a recursively defined FSA, some transitions map to ϵ , the empty move, and the resulting parametric FSA is neither deterministic nor minimum. It was confirmed that a similar recursive formula for a language equivalent ϵ -free parametric FSA can be derived from the recursive formula for the parametric reachability graph, demonstrating that similar structural regularities found in the family of reachability graphs occur after ϵ removal. Furthermore, it was shown that the equivalent deterministic parametric FSA can also be represented by a recursive formula. One difference between the work in [58] and our work is that, for the lossy service case, [58] develops parametric reachability graphs and finite state automata that are defined recursively, whereas in our work the parametric reachability graph and finite state automaton presented in Chapters 5 and 6 are defined using closed-form algebraic expressions, not recursively defined expressions (although closed-form expressions are derived later). A key difference is that the epsilon removal step is performed explicitly, rather than being combined with determination through the use of epsilon closures (as described in Chapter 2). Another key difference is that, as with the work in [40], the work in [58] deals with a service specification and not a protocol specification, and that no attempt is made to verify the CES protocol against the service. Finally, our work has involved *two* protocol parameters, neither of which are the single medium capacity parameter.

3.6 Our own Previous Work

This section summarises our own previous work, including that which is not included in [36–39].

In our previous work [15] we summarised a protocol verification methodology based on CPNs [46] and finite state automata. This methodology uses state space methods and has been applied successfully for finite state systems, for small values of parameters. Techniques such as partial orders, symmetry and equivalence, and packed state spaces [31, 64, 81], and more recently the sweep-line method [16, 27, 53, 61], for alleviating the state space explosion problem [81] help to extend the method to larger ranges of parameters, but cannot handle large or unbounded values.

In [14, 15], the methodology is illustrated using a Stop-and-Wait Protocol model with two parameters: the maximum sequence number, **MaxSeqNo**; and the maximum number of retransmissions, **MaxRetrans**. This model resembles the model presented in Chapter 4. From a modelling point of view, the values of these parameters may be chosen arbitrarily. We would thus like to prove that the SWP class is correct for any values of $\text{MaxSeqNo} \geq 1$ and $\text{MaxRetrans} \geq 0$. This becomes impossible using finite state techniques, as we need to consider an infinite number of increasingly larger finite state spaces. For FIFO channels (either lossy or lossless), a hand proof in [15] shows that the number of messages in the message channel (and the number of acks in the acknowledgement channel) has a least upper bound of $2\text{MaxRetrans} + 1$, for any positive value of **MaxSeqNo**, and any non-negative value of **MaxRetrans**. For reordering channels, a hand proof of the unboundedness of the message and acknowledgement channels was developed in [13–15] using the semantic model of High-level Petri Nets [42]. For other properties, such as that the protocol conforms to its service

of alternating send and receive events, the standard methodology was used to verify this property for selected parameter values in the range $0 < \text{MaxSeqNo} < 1024$, $0 \leq \text{MaxRetrans} \leq 4$, but no general result was obtained, for either lossless channels [15] or lossy channels [13, 15].

In [17, 18] we have used a symbolic verification tool called FAST (Fast Acceleration of Symbolic Transition systems) [32] to analyse the SWP. Like TReX, FAST also uses accelerations (meta-transitions) to perform symbolic analysis of infinite state systems, to encode an arbitrary number of iterations of sequences of actions within the system. Models are represented in FAST using *counter systems*. These are automata extended with vectors of integer variables and whose transitions are labelled with Presburger-linear functions. In essence, FAST views the problem as a labelled transition system that uses tuples of integers to represent the state and *Presburger functions* to represent changes of state [7, 33]. In [18] a methodology was described for converting CPNs into counter systems and the SWP CPN model from [13, 15] was revised to make it easier to convert to a counter system and analyse using FAST. We were able to symbolically verify a number of properties of our SWP model for arbitrary MaxRetrans and for MaxSeqNo from 1 to 5 in [17, 18], with the results extended to arbitrary MaxSeqNo but with $\text{MaxRetrans}=0$ in [17]. The verified properties included: conformance to the property of alternating send and receive events; in-sequence delivery of data; no loss or duplication of data; no deadlocks; and a lowest upper bound on the total number of messages and acknowledgements of $2\text{MaxRetrans}+1$ thus validating the hand proof given in [15] for this range of parameter values. A further result presented in [17] was an automatic proof of the form of the markings of the parametric reachability graph (the algebraic expressions for the markings) as presented in [36, 37], in MaxSeqNo only, with $\text{MaxRetrans}=0$. This complemented the manual proof already performed in [36, 37].

Progress was made toward parametric verification of the Stop-and-Wait Protocol (prior to and concurrently with [17, 18]) in [36, 37], when considering the SWP with no retransmissions. Algebraic expressions representing the family of reachability graphs were developed in the MaxSeqNo parameter only, (i.e. with $\text{MaxRetrans}=0$). These expressions for the parametric reachability graph in MaxSeqNo were then used to parametrically verify the conformance of the SWP to its service of alternating send and receive events, for all values of MaxSeqNo .

Further progress was made in [38, 39], in which some of the core results of [35] were presented. The algebraic expressions for the parametric reachability graph of the SWP were extended to incorporate both the MaxSeqNo and MaxRetrans parameters. Incorporation of the MaxRetrans parameter increased the complexity of the expressions considerably. As mentioned in Chapter 1, an indication of the increase in complexity can be gauged by the size of the reachability graphs: the number of nodes and arcs grow linearly in the MaxSeqNo parameter but quartically in the MaxRetrans parameter. Verification of a number of properties from the parametric reachability graph were sketched in [38] and carried out in [39]. However, neither [38] nor [39] proved conformance of the SWP to its service of alternating send and receive events in both parameters, as we do in this report.

Chapter 4

The Stop-and-Wait Protocol CPN Model

The SWP is modelled using Coloured Petri nets [46,47,52], a form of Petri net in which tokens are arbitrarily complex data values. This model is the same as the one presented in [35,38,39], a variant of the model presented in [36,37]. The following description is based on text from [38]. For a more detailed description of the model, the reader is referred to [35,39].

The CPN diagram is shown in Fig. 4.1 along with all the declarations used in the inscriptions of the CPN diagram in Fig. 4.2. The software tool, Design/CPN [29], was used to produce these figures. The inscription language is a variant of Standard ML [71]. The two parameters `MaxRetrans` and `MaxSeqNo` can be seen at the top of the declarations in Fig. 4.2.

The channels are modelled as lists manipulated by the arc inscriptions as First-In-First-Out (FIFO) queues in places `mess_channel` and `ack_channel`. Transitions `mess_loss` and `ack_loss` model loss, both in the network (buffer overflow in a router) and by discarding messages and acknowledgements with transmission errors (checksum failures). Loss can occur anywhere in the message and acknowledgement queues, not just from the head. This is done via nondeterministic binding of variables `sn` and `rn` and the function `Contains` in the guard of each loss transition, to ensure that `sn` and `rn` are only bound to values that are present in the channels. The removal of the message is via function `LOSS` in the arc inscriptions.

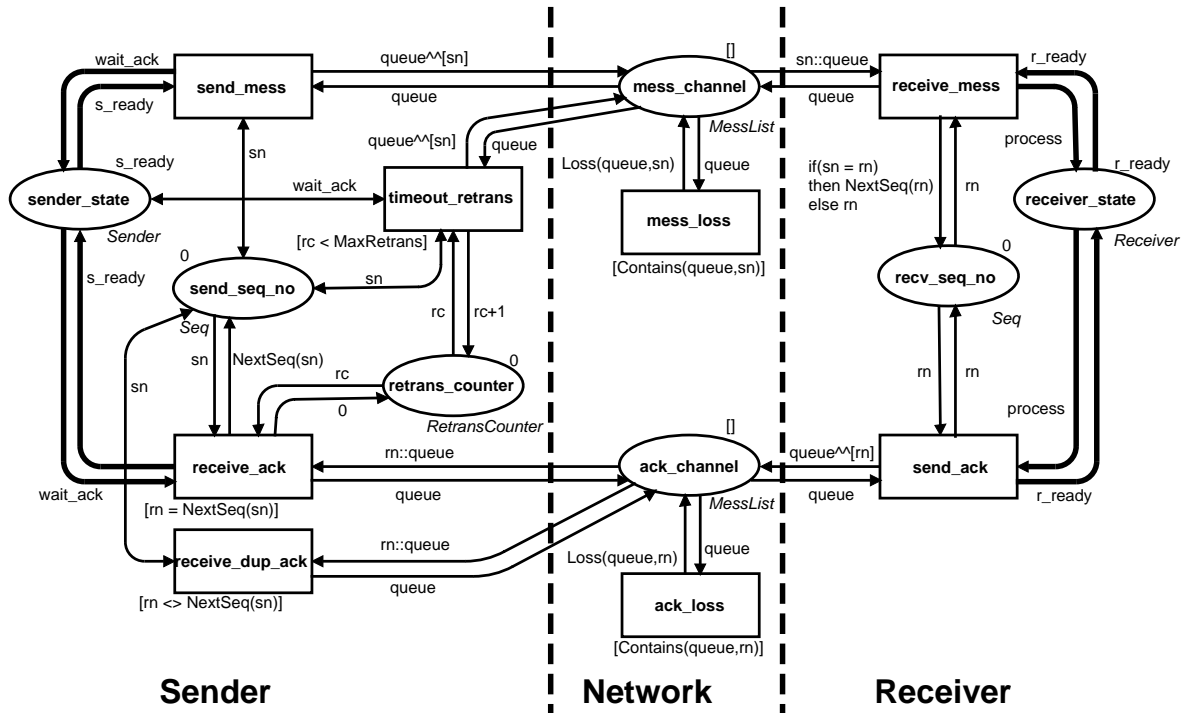


Figure 4.1: A CPN of the Stop-and-Wait Protocol operating over an in-order medium.

```

val MaxRetrans = 0;
val MaxSeqNo = 1;

color Sender = with s_ready | wait_ack;
color Receiver = with r_ready | process;
color Seq = int with 0..MaxSeqNo;
color RetransCounter = int with 0..MaxRetrans;
color Message = Seq;
color MessList = list Message;

var sn,rn : Seq;
var rc : RetransCounter;
var queue : MessList;

fun NextSeq(n) = if(n = MaxSeqNo) then 0 else n+1;
fun Contains([],sn) = false
  | Contains(m::queue,sn) = if(sn=m) then true else Contains(queue,sn);
fun Loss(m::queue,sn) = if(sn=m) then queue else m::Loss(queue,sn);

```

Figure 4.2: Declarations of the CPN shown in Fig. 4.1.

Chapter 5

Parametric Reachability Graph

In this chapter we present the parametric reachability graph of our infinite family of Stop-and-Wait Protocols. This chapter presents the work from [35], a substantially more mature version of the work originally presented in [38, 39]. We begin in Section 5.1 by introducing the notational conventions used in the marking and arc notation developed in Section 5.2. In Section 5.3 we take a closer look at the markings we wish to represent, and define a set of *classes* of marking, which we take advantage of in the definition of our parametric marking and arc notation in Section 5.4. We then establish formulas for determining special sets of markings called *downward-closed* sets, directly from the parametric notation, in Section 5.5. Finally, the parametric reachability graph is presented in Section 5.6.

5.1 Notational Conventions

In the following work, we shall use MS and MR as shorthand for the parameters `MaxSeqNo` and `MaxRetrans` respectively. In this report, the notation $[M\rangle$ is used to represent all markings reachable from marking M . The notation $M[be\rangle$ is used to indicate that the binding element, be , is enabled by marking M .

We can define a Reachability Graph (RG) as follows:

Definition 18 (Reachability Graph). *The **RG** of a CPN with initial marking M_0 and a set of binding elements BE , is a labelled directed graph $RG = (V, A)$ where*

1. $V = [M_0\rangle$ is the set of reachable markings of the CPN; and
2. $A = \{(M, (t, b), M') \in V \times BE \times V \mid M[(t, b)\rangle M'\}$ is the set of labelled directed arcs, where $M[(t, b)\rangle M'$ denotes that the marking of the CPN changes from M to M' on the occurrence of binding element $(t, b) \in BE$, where t is the name of a transition and b is a binding of the variables of the transition to values.

The Stop-and-Wait protocol CPN in Figs. 4.1 and 4.2 is parameterised by `MaxSeqNo` and `MaxRetrans`. We explicitly denote this parameterised CPN and its RG by $CPN_{(MS, MR)}$ and $RG_{(MS, MR)} = (V_{(MS, MR)}, A_{(MS, MR)})$ where $MS = \text{MaxSeqNo}$ and $MR = \text{MaxRetrans}$.

In addition, the following notational conventions are used throughout the remainder of this report:

- i^j is used to represent j repetitions of the message (or acknowledgement) with sequence number i in the message (or acknowledgement) channel;
- \oplus_{MS} is used to represent modulo $MS + 1$ addition; and
- \ominus_{MS} is used to represent modulo $MS + 1$ subtraction.

5.2 Marking and Arc Notation

In order to introduce our notation, it is necessary to prove that every marking of every place is a singleton multiset, in all reachable markings of the CPN. We can do so from the structure of the CPN model:

Lemma 1. *For all reachable markings of $CPN_{(MS,MR)}$, each place in the CPN diagram contains exactly one token, i.e. $\forall M \in V_{(MS,MR)}, |M(\text{sender_state})| = |M(\text{receiver_state})| = |M(\text{send_seq_no})| = |M(\text{recv_seq_no})| = |M(\text{mess_channel})| = |M(\text{ack_channel})| = |M(\text{retrans_counter})| = 1$.*

Proof. Proof is by direct inspection of the CPN in Figs. 4.1 and 4.2. Every transition in the CPN diagram in Fig. 4.1 has both an incoming arc and an outgoing arc to every place with which it interacts. This net structure is independent of the values of the parameters. Each incoming arc removes exactly one token and each outgoing arc produces exactly one token. The value of the token may be affected by the parameter values (i.e. the `MaxSeqNo` parameter, through function `NextSend` defined in the declarations in Fig. 4.2) but the parameter values do not affect the number of tokens removed or produced. We systematically examine each of the places below.

sender_state $|M_0(\text{sender_state})| = |1's_ready| = 1$, hence this lemma holds for the initial marking. The marking of this place can only be changed by the `send_mess`, `timeout_retrans` and `receive_ack` transitions. The occurrence of these transitions either replaces one value by another (the `send_mess` and `receive_ack` transitions) or does not change the marking (the `timeout_retrans` transition). Hence $\forall M \in [M_0], |M(\text{sender_state})| = 1$.

receiver_state $|M_0(\text{receiver_state})| = |1'r_ready| = 1$, hence this lemma holds for the initial marking. The marking of this place can only be changed by the `receive_mess` and `send_ack` transitions. The occurrence of either of these transitions replaces one value by another. Hence $\forall M \in [M_0], |M(\text{receiver_state})| = 1$.

send_seq_no $|M_0(\text{send_seq_no})| = |1'0| = 1$, hence this lemma holds for the initial marking. The marking of this place can only be changed by the `send_mess`, `timeout_retrans`, `receive_ack` and `receive_dup_ack` transitions. The occurrence of these transitions either replaces one value by another (the `receive_ack` transition) or does not change the marking (the `send_mess`, `timeout_retrans` and `receive_dup_ack` transitions). Hence $\forall M \in [M_0], |M(\text{send_seq_no})| = 1$.

recv_seq_no $|M_0(\text{recv_seq_no})| = |1'0| = 1$, hence this lemma holds for the initial marking. The marking of this place can only be changed by the `receive_mess` and `send_ack` transitions. The occurrence of these transitions either replaces one value by another (the `receive_mess` transition in the case of receiving a new message) or does not change the marking (the `receive_mess` transition in the case of receiving a duplicate message and the `send_ack` transition). Hence $\forall M \in [M_0], |M(\text{recv_seq_no})| = 1$.

mess_channel $|M_0(\text{mess_channel})| = |1'\square| = 1$, hence this lemma holds for the initial marking. The marking of this place can only be changed by the `send_mess`, `timeout_retrans`, `mess_loss` and `receive_mess` transitions. The occurrence of any of these transitions replaces one value by another. Hence $\forall M \in [M_0], |M(\text{mess_channel})| = 1$.

ack_channel $|M_0(\text{ack_channel})| = |1'\square| = 1$, hence this lemma holds for the initial marking. The marking of this place can only be changed by the `send_ack`, `ack_loss`, `receive_ack` and `receive_dup_ack` transitions. The occurrence of any of these transitions replaces one value by another. Hence $\forall M \in [M_0], |M(\text{ack_channel})| = 1$.

retrans_counter $|M_0(\text{retrans_counter})| = |1'0| = 1$, hence this lemma holds for the initial marking. The marking of this place can only be changed by the `timeout_retrans` and `receive_ack` transitions. The

occurrence of these transitions either replaces one value by another, or does not change the marking (when no retransmissions of the currently outstanding message have occurred). Hence $\forall M \in [M_0], |M(\text{retrans_counter})| = 1$.

Hence, this property holds for all places in all reachable markings, thus the lemma is proved. \square

We can define a function that converts a singleton multiset to its basis element:

Definition 19 (Singleton Multiset to Element (based on [40]).

Let B_{MS_1} be the set of all singleton multisets over a basis set $A : B_{MS_1} = \{\{(a, 1)\} \mid a \in A\}$. A function that converts a singleton multiset to its basis element is given by $f_c : B_{MS_1} \rightarrow A$, where $f_c(\{(a, 1)\}) = a$.

A corollary of Lemma 1 is that f_c can be applied to the marking of any place in any reachable marking of $CPN_{(MS,MR)}$. Given Lemma 1, we can develop a notation for markings that can be used to uniquely identify every marking in $RG_{(MS,MR)}$ for all values of the parameters.

5.2.1 Marking Notation

A marking of a CPN can be represented as a set of pairs, with one pair for each place, as:

$$M = \{(p, mp) \mid p \in P\}$$

where $mp \in \mu Type(p)$, $Type(p)$ is the type (colour set) of place p , and $\mu Type(p)$ is the set of multisets over the type of place p .

In [35] this set representation was derived from the definition of a High-level Petri Net Graph [42], but for the purposes of this report it suffices to understand informally that the marking of our SWP CPN can be represented as a set of pairs.

For the case where $|mp| = 1$, i.e. $mp = 1'g$ where $g \in Type(p)$, we have a singleton multiset. It is possible to represent a singleton multiset by its basis element, e.g. $1'g$ can be represented by g . Lemma 1 guarantees that for all reachable markings, every place in our SWP CPN model is marked by a singleton multiset. Hence, this allows us to represent a marking of our system by a simpler set of pairs:

$$M = \{(p, g) \mid p \in P\}, \text{ where } g \in Type(p)$$

This set of pairs contains exactly one pair for each place, and hence a marking can be represented by a vector of the markings of each of the places of the net:

$$M = (M(\text{sender_state}), M(\text{receiver_state}), M(\text{send_seq_no}), M(\text{recv_seq_no}), \\ M(\text{mess_channel}), M(\text{ack_channel}), M(\text{retrans_counter})) \quad (5.1)$$

Because of Lemma 1, $M(\text{sender_state})$, $M(\text{receiver_state})$, $M(\text{send_seq_no})$, $M(\text{recv_seq_no})$, and $M(\text{retrans_counter})$ can be represented simply by the value of the token in each of their respective places. We can introduce variables that run over the values of these tokens:

$$\begin{aligned} s_state &\in \text{Sender}, \\ r_state &\in \text{Receiver}, \\ ssn, rsn &\in \text{Seq}, \text{ and} \\ ret &\in \text{RetransCounter}, \text{ respectively.} \end{aligned}$$

The marking of the message channel, $M(\text{mess_channel})$, and acknowledgement channel, $M(\text{ack_channel})$, cannot be represented as easily. In [17, 18] the marking of the message and acknowledgement channels were each encoded into four integer variables. In this report we present a similar encoding for the message and acknowledgement channel markings, but using only two integer variables for each, as well as

knowledge of the sender sequence number. Consider the possible content of the message channel. Intuitively, it may contain zero or more instances of the currently outstanding message as well as zero or more instances of retransmissions of the previous message. Similarly, the acknowledgement channel may contain zero or more instances of the acknowledgement for the currently outstanding message, as well as zero or more instances of old duplicate acknowledgements of the previous message. We state the following lemma regarding the content of the message channel.

Lemma 2. *The content of the message channel can be represented by:*

$$M(\text{mess_channel}) = 1'[(ssn \ominus_{MS} 1)^{mo} ssn^{mn}]$$

where $mo, mn \in \mathbb{N}$ are the number of instances of the previous (old) message (with sequence number $ssn \ominus_{MS} 1$) and currently outstanding (new) message (with sequence number ssn), respectively.

Proof. Consider that we start with an empty message channel. The maximum number of messages with a given sequence number n that can be inserted into the message queue is the original (send_mess occurs) plus MaxRetrans duplicates (by MaxRetrans occurrences of timeout_retrans) giving $M(\text{mess_channel}) = 1'[n^{MR+1}]$. At this point the sender must stop and wait until it receives an acknowledgement (receive_ack) for the n -message. The minimum number of n -messages that need to be received and acknowledged (i.e. when no loss occurs) is one, leaving MaxRetrans n -messages in the message queue. When this acknowledgement is received, the retransmission counter is reset to zero and the n -messages that were considered 'new' are now considered 'old' because the sender sequence number has incremented to $n \oplus_{MS} 1$. At this point, (MaxRetrans+1) $(n \oplus_{MS} 1)$ -messages can be sent, giving $M(\text{mess_channel}) = 1'[n^{MR} (n \oplus_{MS} 1)^{MR+1}]$. Because of the FIFO property of the communication channels, the remaining MaxRetrans n -messages must be removed (by loss or receipt) before the first $(n \oplus_{MS} 1)$ -message can be received and acknowledged, allowing messages with sequence number $n \oplus_{MS} 2$ to be placed in the channel. Thus before any new message can be sent, there can only be messages with a single sequence number (the 'old' sender sequence number, $ssn \ominus_{MS} 1$) in the channel. Hence the channel content can be represented by $[(ssn \ominus_{MS} 1)^{mo} ssn^{mn}]$ and the lemma is proved. \square

Similarly, for the acknowledgement channel:

Lemma 3. *The content of the acknowledgement channel can be represented by:*

$$M(\text{ack_channel}) = 1'[ssn^{ao} (ssn \oplus_{MS} 1)^{an}]$$

where $ao, an \in \mathbb{N}$ are the number of instances of the acknowledgement of the previous message and the currently outstanding message, respectively.

Proof. The proof is similar to that of Lemma 2, except that we are dealing with acknowledgements in the acknowledgement channel, not messages in the message channel, with sequence numbers of ssn and $ssn \oplus_{MS} 1$ instead of $ssn \ominus_{MS} 1$ and ssn , respectively. \square

In [35, 39] it was proved that this representation is sufficiently expressive to capture the channel content of all reachable markings. Thus from Equation (5.1), the marking of the net can be represented by the vector

$$M = (s_state, r_state, ssn, rsn, [(ssn \ominus_{MS} 1)^{mo} ssn^{mn}], [ssn^{ao} (ssn \oplus_{MS} 1)^{an}], ret)$$

and hence we can see that a marking of the CPN can be characterised by 9 variables, plus the two parameters.

We are now ready to define our marking notation.

Definition 20 (Marking Notation).

$M_{(s_state, r_state), (ssn, rsn), (mo, ao, mn, an, ret)}^{(MS, MR)}$ represents a marking of the SWP CPN model from Figs. 4.1 and 4.2, where the superscript is a pair of the parameter values of the SWP CPN and the subscript encodes the marking description, such that:

- $s_state \in \text{Sender}$ is the state of the Sender, i.e. either ready or waiting for an acknowledgement;
- $r_state \in \text{Receiver}$ is the state of the Receiver, i.e. either ready or processing a message;
- $ssn \in \text{Seq}$ is the sender sequence number;
- $rsn \in \text{Seq}$ is the receiver sequence number;
- $mo \in \mathbb{N}$ is the number of old messages in the message channel, i.e. the number of instances of the previously acknowledged message (with a sequence number equal to the previous sender sequence number, $ssn \ominus_{MS} 1$) in the message channel;
- $ao \in \mathbb{N}$ is the number of old acknowledgements in the acknowledgement channel, i.e. the number of instances of the acknowledgement of the previous message (acknowledgements with a sequence number equal to the sender sequence number, ssn) in the acknowledgement channel;
- $mn \in \mathbb{N}$ is the number of instances of the new (or **current**) message in the message channel, i.e. the number of instances of the currently outstanding message (with a sequence number equal to the sender sequence number, ssn) in the message channel;
- $an \in \mathbb{N}$ is the number of new (or current) acknowledgements in the acknowledgement channel, i.e. the number of instances of the acknowledgement of the currently outstanding message (acknowledgements with a sequence number equal to the next sender sequence number, $ssn \oplus_{MS} 1$) in the acknowledgement channel; and
- $ret \in \text{RetransCounter}$ is the number of times the currently outstanding message has been retransmitted;

where the marking of each place is given by:

$$\begin{aligned}
 M(\text{sender_state}) &= 1's_state & M(\text{receiver_state}) &= 1'r_state \\
 M(\text{send_seq_no}) &= 1'ssn & M(\text{recv_seq_no}) &= 1'rsn \\
 M(\text{retrans_counter}) &= 1'ret \\
 M(\text{mess_channel}) &= 1'[(ssn \ominus_{MS} 1)^{mo} ssn^{mn}] \\
 M(\text{ack_channel}) &= 1'[ssn^{ao} (ssn \oplus_{MS} 1)^{an}]
 \end{aligned}$$

In Definition 20 we retain the square brackets around the content of the message and acknowledgement channels to remind us that the channel content is represented as a list, rather than a string, in the CPN model.

The motivation behind the particular grouping of the subscript variables is an attempt to reflect one logical grouping that could be made, dividing the variables into: 1) sender and receiver state; 2) sender and receiver sequence number; and 3) channel content description. The retransmission counter is grouped into the channel content description because of its intimate relationship with the channel content.

As an example, consider the marking represented by $M_{(wait_ack,r_ready),(2,2),(2,1,2,0,1)}^{(3,4)}$. This corresponds to the marking of $CPN_{(3,4)}$:

$$\begin{aligned}
 M(\text{sender_state}) &= 1'wait_ack & M(\text{receiver_state}) &= 1'r_ready \\
 M(\text{send_seq_no}) &= 1'2 & M(\text{recv_seq_no}) &= 1'2 \\
 M(\text{mess_channel}) &= 1'[1, 1, 2, 2] & M(\text{ack_channel}) &= 1'[2] \\
 M(\text{retrans_counter}) &= 1'1
 \end{aligned}$$

where $MS = 3$ and $MR = 4$.

5.2.2 Arc Notation

We now look to define arc notation analogously. Each arc in $RG_{(MS,MR)}$ takes the form $(M, (t, b), M')$ where M and M' are markings in $V_{(MS,MR)}$ and $(t, b) \in BE$. The state information of the source marking, M , along with information about the binding element, (t, b) , allows every arc in $RG_{(MS,MR)}$ to be uniquely identified. This follows from the definition of the arcs of an RG in Definition 18 and the transition rule for CPNs [46] as the marking of the CPN changes from the source marking to the destination marking on the firing of the enabled binding element.

In many cases, the binding of variables that will enable a specific transition in a marking can be determined from that marking. For example, consider the `send_mess` transition and a marking, $M_{(s_ready, r_ready), (0,0), (0,0,0,0,0)}$, that enables `send_mess`. From the enabling rules of CPNs [46], `send_mess` will only be enabled in this marking if the variable sn is bound to 0 and $queue$ is bound to [], which we can determine by looking at the source marking.

Pursuing this idea, we examine the binding of variables of each transition, $t \in T$, given a reachable marking, $M = M_{(s_state, r_state), (ssn, rsn), (mo, ao, mn, an, ret)}$, such that $M[t]$ (t is enabled in M for some binding of its variables). In [39] it was demonstrated that the enabling of each transition and its subsequent firing is invariant with respect to the concrete values of the sequence numbers involved, provided that the relative values of all sequence numbers involved is preserved. This was formalised in the context of HLPNG's in [35]. Hence, we can state the following:

- $M[\text{send_mess}]$ with the binding $(queue = [(ssn \ominus_{MS} 1)^{mo} ssn^{mn}], sn = ssn)$.
- $M[\text{timeout_retrans}]$ with the binding $(queue = [(ssn \ominus_{MS} 1)^{mo} ssn^{mn}], sn = ssn, rc = ret)$, with $rc < MR$.
- $M[\text{receive_mess}]$ with the binding:
 - $(queue = [(ssn \ominus_{MS} 1)^{mo-1} ssn^{mn}], sn = ssn \ominus_{MS} 1, rn = rsn)$, if $mo > 0$; or
 - $(queue = [ssn^{mn-1}], sn = ssn, rn = rsn)$, if $mo = 0$ and $mn > 0$.

These two possible enabled bindings are due to the fact that we may or may not have old messages in the channel, and our marking notation specifies that old messages come before new messages in the (FIFO) channel.

- $M[\text{send_ack}]$ with the binding $(queue = [ssn^{ao} (ssn \oplus_{MS} 1)^{an}], rn = rsn)$.
- $M[\text{receive_ack}]$ with the binding $(queue = [(ssn \oplus_{MS} 1)^{an-1}], sn = ssn, rn = ssn \oplus_{MS} 1, rc = ret)$. Recall that rn must equal $sn \oplus_{MS} 1$ from the guard. The guard prevents this transition from being enabled if there are any old acknowledgements in the channel (i.e. ao must equal zero) because our marking notation specifies that old acknowledgements come before new acknowledgements in the (FIFO) channel.
- $M[\text{receive_dup_ack}]$ with the binding $(queue = [ssn^{ao-1} (ssn \oplus_{MS} 1)^{an}], sn = ssn, rn = ssn)$. Recall that rn cannot equal $sn \oplus_{MS} 1$ from the guard. Our marking notation specifies that old acknowledgements have sequence number ssn and that the number of old acknowledgements is given by ao . Hence, ao must be greater than zero for this transition to be enabled.
- $M[\text{mess_loss}]$ with the binding:
 - $(queue = [(ssn \ominus_{MS} 1)^{mo} ssn^{mn}], sn = ssn \ominus_{MS} 1)$, provided $mo > 0$; or
 - $(queue = [(ssn \ominus_{MS} 1)^{mo} ssn^{mn}], sn = ssn)$, provided $mn > 0$.

These two possible enabled bindings arise because loss of a message can occur from anywhere in the message channel, hence either an old message or a new message can be lost.

- $M[\text{ack_loss}]$ with the binding:

- $(\text{queue} = [ssn^{ao} (ssn \oplus_{MS} 1)^{an}], rn = ssn)$, provided $ao > 0$; or
- $(\text{queue} = [ssn^{ao} (ssn \oplus_{MS} 1)^{an}], rn = ssn \oplus_{MS} 1)$, provided $an > 0$.

These two possible enabled bindings arise because, like messages, loss of an acknowledgement can occur from anywhere in the acknowledgement channel, hence either an old or a new acknowledgement can be lost.

Hence, for transitions `send_mess`, `timeout_retrans`, `receive_mess`, `send_ack`, `receive_dup_ack`, and `receive_ack`, we can reconstruct the corresponding enabled binding element from the source marking and transition name only. (In the case of `receive_mess`, the specific enabled binding element is determined by the value of mo in the source marking.)

However, as can be seen, this is not the case for the `mess_loss` or `ack_loss` transitions. Taking `mess_loss` as an example, if there are both old messages ($mo > 0$) and new messages ($mn > 0$) in the message channel, then it is not possible to determine from the transition name and source marking whether an occurrence of `mess_loss` will result in loss of an old message or loss of a new message. The same situation exists for the `ack_loss` transition. For these two transitions, one extra piece of information is needed to be able to reconstruct the corresponding enabled binding element: whether it is an old message (or acknowledgement) or a new message (or acknowledgement) that is being lost. We differentiate these two cases by appending `old` or `new` to the transition name. Hence, we can define a set of augmented transition names, $ATNames$, where:

$$ATNames = \{\text{send_mess}, \text{timeout_retrans}, \text{receive_mess}, \text{send_ack}, \text{receive_ack}, \\ \text{receive_dup_ack}, \text{mess_loss_old}, \text{mess_loss_new}, \text{ack_loss_old}, \text{ack_loss_new}\}$$

that encode the enabled binding elements for a source marking, given that the corresponding transition is enabled in that source marking. A simple surjective mapping, $TransMap$, can be defined to map from augmented transition names to transitions:

Definition 21 (Augmented Transition Name to Transition Name).

A surjective mapping, $TransMap : ATNames \rightarrow T$, from augmented transition names to transition names, is given by:

$$TransMap(atn) = \begin{cases} atn, & \text{for } atn \in \{\text{send_mess}, \text{timeout_retrans}, \text{receive_mess}, \text{send_ack}, \\ & \text{receive_ack}, \text{receive_dup_ack}\}, \\ \text{mess_loss}, & \text{for } atn \in \{\text{mess_loss_old}, \text{mess_loss_new}\}, \\ \text{ack_loss}, & \text{for } atn \in \{\text{ack_loss_old}, \text{ack_loss_new}\}. \end{cases}$$

The mapping from source markings and $ATNames$ to enabled binding elements is defined formally below.

Definition 22 (Reconstruct Enabled Binding Element).

Given a marking, M , that enables a transition, $t \in T$, a function, $ReconstructBindingElement$, which maps from M and an augmented transition name (whose corresponding transition is enabled in M) to the corresponding enabled binding element, according to Table 5.1, is given by

$$ReconstructBindingElement : V_{subset} \times ATNames \rightarrow BE$$

where $V_{subset} \times ATNames = \{(M, atn) \mid M[TransMap(atn)]\}$ and

$$ReconstructBindingElement(M, atn) = (t, b)$$

where (t, b) is a binding element from column 2 of Table 5.1 corresponding to the source marking M and an augmented transition name from column 1 of Table 5.1.

Table 5.1: A mapping from an augmented transition name, atn , to its corresponding enabled binding element, given a source marking, $M = M_{(s_state,r_state),(ssn,rsn),(mo,ao,mn,an,ret)}^{(MS,MR)}$, that enables $TransMap(atn)$.

Augmented Transition Name, $atn \in ATNames$	Corresponding Enabled Binding Element, $ReconstructBindingElement(M, atn)$.
send_mess	$(send_mess, (queue = [(ssn \ominus_{MS} 1)^{mo} ssn^{mn}], sn = ssn))$.
timeout_retrans	$(timeout_retrans, (queue = [(ssn \ominus_{MS} 1)^{mo} ssn^{mn}], sn = ssn, rc = ret))$.
receive_mess	$(receive_mess, (queue = [(ssn \ominus_{MS} 1)^{mo-1} ssn^{mn}], sn = ssn \ominus_{MS} 1, rn = rsn))$, provided $mo > 0$; or $(receive_mess, (queue = [ssn^{mn-1}], sn = ssn, rn = rsn))$ provided $mo = 0$ and $mn > 0$.
send_ack	$(send_ack, (queue = [ssn^{ao} (ssn \oplus_{MS} 1)^{an}], rn = rsn))$.
receive_ack	$(receive_ack, (queue = [(ssn \oplus_{MS} 1)^{an-1}], sn = ssn, rn = ssn \oplus_{MS} 1, rc = ret))$.
receive_dup_ack	$(receive_dup_ack, (queue = [ssn^{ao-1} (ssn \oplus_{MS} 1)^{an}], sn = ssn, rn = ssn))$.
mess_loss_old	$(mess_loss, (queue = [(ssn \ominus_{MS} 1)^{mo} ssn^{mn}], sn = ssn \ominus_{MS} 1))$, provided $mo > 0$.
mess_loss_new	$(mess_loss, (queue = [(ssn \ominus_{MS} 1)^{mo} ssn^{mn}], sn = ssn))$, provided $mn > 0$.
ack_loss_old	$(ack_loss, (queue = [ssn^{ao} (ssn \oplus_{MS} 1)^{an}], rn = ssn))$, provided $ao > 0$.
ack_loss_new	$(ack_loss, (queue = [ssn^{ao} (ssn \oplus_{MS} 1)^{an}], rn = ssn \oplus_{MS} 1))$, provided $an > 0$.

We are now ready to define our arc notation.

Definition 23 (Arc Notation).

$a_{atn,(s_state,r_state),(ssn,rsn),(mo,ao,mn,an,ret)}^{(MS,MR)}$ represents an arc, $(M, (t, b), M')$, in $RG_{(MS,MR)}$, with

- source marking, $M = M_{(s_state,r_state),(ssn,rsn),(mo,ao,mn,an,ret)}^{(MS,MR)}$;
- binding element, $(t, b) = ReconstructBindingElement(M, atn)$; and
- destination marking, M' , given by $M[(t, b)]M'$, where M' is given by the transition rule.

As an example of arc notation, the arc identified by $a_{send_mess,(s_ready,r_ready),(2,2),(1,1,0,0,0)}^{(3,4)}$ is the arc, $(M, (t, b), M')$, where:

$$M = M_{(s_ready,r_ready),(2,2),(1,1,0,0,0)}^{(3,4)},$$

$$(t, b) = (send_mess, (queue = [1], sn = 2)), \text{ and}$$

$$M' = M_{(wait_ack,r_ready),(2,2),(1,1,1,0,0)}^{(3,4)}$$

which corresponds to the occurrence of the `send_mess` transition, with binding $(queue = [1], sn = 2)$, from marking $M_{(s_ready,r_ready),(2,2),(1,1,0,0,0)}^{(3,4)}$, i.e. the sender is sending a message with sequence number 2 from a marking in which the sender and receiver are both ready, the receiver is expecting a message with sequence number 2, there is one old message (with sequence number 1) in the message channel, and there is one old acknowledgement (with sequence number 2) in the acknowledgement channel.

Note that the notation presented in Definitions 20 and 23 is more expressive than required for the representation of all markings and arcs in $RG_{(MS,MR)}$ because not all relationships between the variables s_state , r_state , ssn and rsn , or between the variables and the parameters MS and MR , have been specified.

5.3 Classifying Markings

When duplicate messages and acknowledgements can exist (i.e. when $\text{MaxRetrans} > 0$) there are more possible combinations of sender and receiver state than when $\text{MaxRetrans}=0$. While it is possible to partition the set of nodes of $RG_{(MS,MR)}$ based on sender and receiver sequence number and an informal concept of the ‘class’ of marking, as was done in [36], it is worth looking more closely at the possible combinations of sender and receiver state for the general case of $\text{MaxRetrans} \geq 0$. This has been done in [39] and more formally in [35].

Table 5.2 captures the possible combinations of, and relationships between, the sender state, receiver state, sender sequence number and receiver sequence number that are missing from the general marking and arc notation in Definitions 20 and 23. Each combination is given a ‘class’ as shown in column 5. There are two possible major states for each of the sender and receiver (shown in columns 1 and 2 respectively) and for two combinations of major state there are two alternatives for sender and receiver sequence number (columns 3 and 4), giving six classes in total. An explanation of each class follows:

- **Class 1:** This may correspond to the initial state, where the sender has yet to transmit the first message and the receiver is ready to receive (sender and receiver are ready, $ssn = rsn = 0$), or to the sender having just received an acknowledgement for the most recent outstanding message while the receiver is ready to receive the next message (sender and receiver are ready, $ssn = rsn = i, 0 \leq i \leq MS$).
- **Class 2:** These are markings where the sender is waiting for an acknowledgement for the currently outstanding message and the receiver is in its ready state (sender is waiting, receiver is ready, $ssn = i$). They may be of one of two sub-classes based on the sequence number of the receiver:
 - **Class 2a:** the receiver has not yet received the currently outstanding message ($rsn = i$); or
 - **Class 2b:** the receiver has received at least one instance of the currently outstanding message ($rsn = i \oplus_{MS} 1$), be it the original or a duplicate, has sent an acknowledgement (that has not yet been received by the sender) and is back in its ready state.
- **Class 3:** These are markings in which the sender is waiting for an acknowledgement for the currently outstanding message and the receiver is currently processing a message (sender is waiting, receiver is processing, $ssn = i$). Such states may be of one of two sub-classes based on the receiver sequence number:
 - **Class 3a:** the message being processed by the receiver is a duplicate of the previously acknowledged message caused by retransmissions at the sender ($rsn = i$); or
 - **Class 3b:** the message being processed by the receiver is the currently outstanding message ($rsn = i \oplus_{MS} 1$).
- **Class 4:** The sender is ready to send a new message but the receiver is processing a duplicate of the previous message (sender is ready, receiver is processing, $ssn = rsn = i$).

As is proved in [35,39], it turns out that all reachable markings of $CPN_{(MS,MR)}$ fall into one of these 6 classes. The classification function can be formally defined as follows.

Definition 24 (Classification Function).

A function, $Class_{MS}$, which classifies the set of reachable markings according to Table 5.2, is given by

$$Class_{MS} : V_{(MS,MR)} \rightarrow \{1, 2a, 2b, 3a, 3b, 4\}$$

which maps each marking, $M \in V_{(MS,MR)}$, to one of the 6 classes given in Table 5.2.

Table 5.2: Classification of markings into Classes of markings based on the state of the sender and receiver.

$M(\text{sender_state})$	$M(\text{receiver_state})$	$M(\text{send_seq_no})$	$M(\text{recv_seq_no})$	$\text{Class}_{MS}(M)$
1's_ready	1'r_ready	1'ssn	1'sn	1
1'wait_ack	1'r_ready	1'ssn	1'sn	2a
1'wait_ack	1'r_ready	1'ssn	$1'sn \oplus_{MS} 1$	2b
1'wait_ack	1'process	1'ssn	1'sn	3a
1'wait_ack	1'process	1'ssn	$1'sn \oplus_{MS} 1$	3b
1's_ready	1'process	1'ssn	1'sn	4

5.4 Parametric Marking and Arc Notation

The classes defined in Table 5.2 provide additional restrictions on the values of the variables in the tuples $(s_state, r_state), (ssn, rsn)$ and also allow the information contained in these tuples to be represented in a more compact form, i.e. by a single pair $(class, ssn)$. This makes the notation less cumbersome to use. Using this more compact representation, we present the definition of our shorthand marking and arc notation. Sets of markings and sets of arcs are also defined using this shorthand notation.

Definition 25 (Shorthand Marking Notation).

A reachable marking, $M \in V_{(MS,MR)}$, is uniquely represented by $M_{(class,ssn),(mo,ao,mn,an,ret)}^{(MS,MR)}$ where the superscript contains the parameter values of the SWP CPN and the subscript contains the marking description, such that $class = \text{Class}_{MS}(M)$, and the parameters MS and MR and the variables ssn, mo, ao, mn, an and ret are as given in Definition 20.

Definition 26 (Shorthand Arc Notation).

An arc, $(M, (t, b), M') \in A_{(MS,MR)}$, is represented by $a_{atn,(class,ssn),(mo,ao,mn,an,ret)}^{(MS,MR)}$ where $class = \text{Class}_{MS}(M)$, and the parameters MS and MR and the variables atn, ssn, mo, ao, mn, an and ret are as given in Definition 23.

Definition 27 (Shorthand Sets of Markings).

1. $V_{(class,ssn)}^{(MS,MR)} = \{M \in V_{(MS,MR)} \mid \text{Class}_{MS}(M) = class, M(\text{send_seq_no}) = 1'ssn\}$ represents the set of markings in which the sender sequence number is given by ssn , and the sender and receiver states and receiver sequence number are given by the class as specified in Table 5.2.
2. $V_{ssn}^{(MS,MR)} = \{M \in V_{(MS,MR)} \mid M(\text{send_seq_no}) = 1'ssn\}$ represents the set of markings in which the sender sequence number is given by ssn .

Definition 28 (Shorthand Sets of Arcs).

1. $A_{(class,ssn)}^{(MS,MR)} = \{(M, (t, b), M') \in A_{(MS,MR)} \mid \text{Class}_{MS}(M) = class, M(\text{send_seq_no}) = 1'ssn\}$ represents the set of arcs where each arc's source node, M , is in $V_{(class,ssn)}^{(MS,MR)}$.
2. $A_{ssn}^{(MS,MR)} = \{(M, (t, b), M') \in A_{(MS,MR)} \mid M(\text{send_seq_no}) = 1'ssn\}$ represents the set of arcs with source nodes in $V_{ssn}^{(MS,MR)}$.

5.5 Downward-Closed Sets of Markings

Using the ideas and terminology from [3] we define the set of *downward-closed* markings of a given reachable marking, M . These are sets containing markings that are identical but for the content of the channels. For

any given marking, M' , in the downward-closed set of M , all identical markings which have as their channel content a substring of the channel content of M' are also present in the set.

To illustrate, consider a marking M where $M(\text{mess_channel}) = 1^c[1, 1, 2]$ and $M(\text{ack_channel}) = 1^c[2]$. The downward-closed set of M thus contains all the markings with every combination of message channel content $\in \{[1, 1, 2], [1, 1], [1, 2], [1], [2], []\}$ and acknowledgement channel content $\in \{[2], []\}$, but with identical markings for all other places (12 markings in all). We formalise this in the following definition.

Definition 29 (Downward-Closed Sets of Markings).

The downward-closed set of a marking, $M_{(class,ssn),(mo,ao,mn,an,ret)}^{(MS,MR)} \in V_{(MS,MR)}$, is given by the function $DownwardClosed : V \rightarrow 2^V$ where:

$$\begin{aligned} DownwardClosed(M_{(class,ssn),(mo,ao,mn,an,ret)}^{(MS,MR)}) \\ = \{M_{(class,ssn),(mo',ao',mn',an',ret)}^{(MS,MR)} \mid 0 \leq mo' \leq mo, 0 \leq ao' \leq ao, 0 \leq mn' \leq mn, 0 \leq an' \leq an\} \end{aligned}$$

The function $DownwardClosed$ can be extended to sets of markings:

Definition 30 (Downward-Closed Set of a Set of Markings).

The downward-closed set of a set of markings, $V_a \in V_{(MS,MR)}$, is given by the function $DC : 2^V \rightarrow 2^V$ where:

$$\begin{aligned} DC(V_a) &= \bigcup_{M \in V_a} DownwardClosed(M) \\ &= \{M_{(class,ssn),(mo',ao',mn',an',ret)}^{(MS,MR)} \mid M_{(class,ssn),(mo,ao,mn,an,ret)}^{(MS,MR)} \in V_a, 0 \leq mo' \leq mo, \\ &\quad 0 \leq ao' \leq ao, 0 \leq mn' \leq mn, 0 \leq an' \leq an\} \end{aligned}$$

We say that a set V_a is downward-closed iff $\forall M \in V_a, DownwardClosed(M) \subseteq V_a$.

Because a downward-closed set of a marking contains exactly those markings which are identical except for having fewer messages and/or acknowledgements in the channels, $DownwardClosed(M)$ can be calculated for $M \in V_{(MS,MR)}$ by firing the `mess_loss` and `ack_loss` transitions repeatedly and in every possible order until both the message and acknowledgement channels are empty.

5.6 The Parametric Reachability Graph

We now specify the markings and arcs of $RG_{(MS,MR)}$ using the notation from Definitions 25, 26, 27 and 28 by specifying allowable ranges of the five variables, $mo, ao, mn, an,$ and ret . These variables can only take non-negative values, i.e. $mo, ao, mn, an, ret \in \mathbb{N}$.

5.6.1 Sets of Markings

All of the markings of $RG_{(MS,MR)}$ are described in Table 5.3, by evaluating the expressions in this table for all $i, 0 \leq i \leq MS$. The first column gives the name of the set of markings in shorthand marking set notation. Column 2 defines the set of markings by specifying the allowable ranges of variable values. If a variable is restricted to a specific value, e.g. 0, we write this directly in the label of the marking. Note that markings of class 3a and class 4 (rows 4 and 6) only exist when $MR > 0$, as is reflected in the condition $0 \leq mo + ao \leq MR - 1$. Hence, $V_{(3a,i)}^{(MS,MR)} = V_{(4,i)}^{(MS,MR)} = \{\}$ when $MR = 0$. We can define the set of all states with the same sender sequence number:

Definition 31 (Markings with the same sender sequence number).

The set, $V_i^{(MS,MR)}$, denotes all markings in $RG_{(MS,MR)}$ with sender sequence number equal to $i, i \in \{0, 1, \dots, MS\}$, where:

$$V_i^{(MS,MR)} = V_{(1,i)}^{(MS,MR)} \cup V_{(2a,i)}^{(MS,MR)} \cup V_{(2b,i)}^{(MS,MR)} \cup V_{(3a,i)}^{(MS,MR)} \cup V_{(3b,i)}^{(MS,MR)} \cup V_{(4,i)}^{(MS,MR)}$$

5.6. The Parametric Reachability Graph

Table 5.3: $V_i^{(MS,MR)} = \bigcup_{class \in Class} V_{(class,i)}^{(MS,MR)}$, for $0 \leq i \leq MS$ and $Class = \{1, 2a, 2b, 3a, 3b, 4\}$.

Name	Set Definition
$V_{(1,i)}^{(MS,MR)}$	$\{M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR\}$
$V_{(2a,i)}^{(MS,MR)}$	$\{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR\}$
$V_{(2b,i)}^{(MS,MR)}$	$\{M_{(2b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1, 0 \leq ret \leq MR\}$
$V_{(3a,i)}^{(MS,MR)}$	$\{M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR\}$
$V_{(3b,i)}^{(MS,MR)}$	$\{M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR\}$
$V_{(4,i)}^{(MS,MR)}$	$\{M_{(4,i),(mo,ao,0,0,0)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1\}$

5.6.2 Sets of Arcs

All of the arcs of $RG_{(MS,MR)}$ are described in Tables 5.4 to 5.9 by evaluating each of these tables for every i , $0 \leq i \leq MS$. There is one table of arcs per row of Table 5.3, describing the set of outgoing arcs of the corresponding set of markings. We can define the set of all arcs with source markings having the same sequence number:

Definition 32 (Arcs with source markings that all have the same sender sequence number).

The set, $A_i^{(MS,MR)}$, denotes the set of arcs with source markings in $V_i^{(MS,MR)}$, where:

$$A_i^{(MS,MR)} = A_{(1,i)}^{(MS,MR)} \cup A_{(2a,i)}^{(MS,MR)} \cup A_{(2b,i)}^{(MS,MR)} \cup A_{(3a,i)}^{(MS,MR)} \cup A_{(3b,i)}^{(MS,MR)} \cup A_{(4,i)}^{(MS,MR)}$$

Each row in each arc table defines a set of arcs, with one row for each transition with at least one enabled binding element in the corresponding set of markings from Table 5.3. The sets $A_{(3a,i)}^{(MS,MR)}$ and $A_{(4,i)}^{(MS,MR)}$ are only defined for $MR > 0$, i.e. when the sets $V_{(3a,i)}^{(MS,MR)}$ and $V_{(4,i)}^{(MS,MR)}$ are not empty. Each table of arcs contains 5 columns. The first column identifies the rows in each table with a row number, for ease of reference. The second column gives the name of the set of arcs being described, in shorthand arc notation. The set of source markings for each row's arcs are not explicitly represented in this table, as they can be easily derived from the shorthand arc notation according to Definition 26 in Section 5.4. The third column gives the binding elements of the arcs defined by each row. The fourth column gives the destination marking of each arc. This source marking can be derived by using the transition rule for CPNs [46]. The fifth column gives additional restrictions to the allowable values of the (mo, ao, mn, an, ret) variables, which already have restrictions based on the corresponding set of source markings. For example, the set of arcs, $A_{(1,i)}^{(MS,MR)}$, shown in Table 5.4 corresponds to all arcs with source nodes in $V_{(1,i)}^{(MS,MR)}$, and so the values of mn, an and ret are equal to zero for all arcs in $A_{(1,i)}^{(MS,MR)}$ due to the restrictions imposed in the definition of $V_{(1,i)}^{(MS,MR)}$.

5.6.3 The Parametric SWP Reachability Graph

We now state the theorem for our parametric reachability graph over both parameters. The correctness of this theorem has been proved in [35, 39].

Theorem 1. For $MS \in \mathbb{N}^+$ and $MR \in \mathbb{N}$, $RG_{(MS,MR)} = (V_{(MS,MR)}, A_{(MS,MR)})$ where

$$V_{(MS,MR)} = \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \text{ and } A_{(MS,MR)} = \bigcup_{0 \leq i \leq MS} A_i^{(MS,MR)}$$

Table 5.4: The set of arcs, $A_{(1,i)}^{(MS,MR)}$, with source markings in $V_{(1,i)}^{(MS,MR)}$.

Row	Arcs	Binding Element	Destination Marking	New Restrictions
1	$a_{\text{send_mess},(1,i),(mo,ao,0,0,0)}^{(MS,MR)}$	$\text{send_mess}\langle\text{queue} = [(i \ominus_{MS} 1)^{mo}], \text{sn}=i\rangle$	$M_{(2a,i),(mo,ao,1,0,0)}^{(MS,MR)}$	none
2	$a_{\text{mess_loss_old},(1,i),(mo,ao,0,0,0)}^{(MS,MR)}$	$\text{mess_loss}\langle\text{queue} = [(i \ominus_{MS} 1)^{mo}], \text{sn}=i \ominus_{MS} 1\rangle$	$M_{(1,i),(mo-1,ao,0,0,0)}^{(MS,MR)}$	$mo \geq 1$
3	$a_{\text{receive_mess},(1,i),(mo,ao,0,0,0)}^{(MS,MR)}$	$\text{receive_mess}\langle\text{queue} = [(i \ominus_{MS} 1)^{mo-1}], \text{sn}=i \ominus_{MS} 1, \text{rn}=i\rangle$	$M_{(4,i),(mo-1,ao,0,0,0)}^{(MS,MR)}$	$mo \geq 1$
4	$a_{\text{ack_loss_old},(1,i),(mo,ao,0,0,0)}^{(MS,MR)}$	$\text{ack_loss}\langle\text{queue} = [i^{ao}], \text{rn}=i\rangle$	$M_{(1,i),(mo,ao-1,0,0,0)}^{(MS,MR)}$	$ao \geq 1$
5	$a_{\text{receive_dup_ack},(1,i),(mo,ao,0,0,0)}^{(MS,MR)}$	$\text{receive_dup_ack}\langle\text{queue} = [i^{ao-1}], \text{sn}=i, \text{rn}=i\rangle$	$M_{(1,i),(mo,ao-1,0,0,0)}^{(MS,MR)}$	$ao \geq 1$

Table 5.5: The set of arcs, $A_{(2a,i)}^{(MS,MR)}$, with source markings in $V_{(2a,i)}^{(MS,MR)}$.

Row	Arcs	Binding Element	Destination Marking	New Restrictions
1	$a_{\text{timeout_retrans},(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$	$\text{timeout_retrans}\langle\text{queue} = [(i \ominus_{MS} 1)^{mo} i^{mn}], \text{sn}=i, \text{rc}=ret\rangle$	$M_{(2a,i),(mo,ao,mn+1,0,ret+1)}^{(MS,MR)}$	$ret < MR$
2	$a_{\text{mess_loss_old},(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$	$\text{mess_loss}\langle\text{queue} = [(i \ominus_{MS} 1)^{mo} i^{mn}], \text{sn}=i \ominus_{MS} 1\rangle$	$M_{(2a,i),(mo-1,ao,mn,0,ret)}^{(MS,MR)}$	$mo \geq 1$
3	$a_{\text{mess_loss_new},(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$	$\text{mess_loss}\langle\text{queue} = [(i \ominus_{MS} 1)^{mo} i^{mn}], \text{sn}=i\rangle$	$M_{(2a,i),(mo,ao,mn-1,0,ret)}^{(MS,MR)}$	$mn \geq 1$
4	$a_{\text{receive_mess},(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$	$\text{receive_mess}\langle\text{queue} = [(i \ominus_{MS} 1)^{mo-1} i^{mn}], \text{sn}=i \ominus_{MS} 1, \text{rn}=i\rangle$	$M_{(3a,i),(mo-1,ao,mn,0,ret)}^{(MS,MR)}$	$mo \geq 1$
5	$a_{\text{receive_mess},(2a,i),(0,ao,mn,0,ret)}^{(MS,MR)}$	$\text{receive_mess}\langle\text{queue} = [i^{mn-1}], \text{sn}=i, \text{rn}=i\rangle$	$M_{(3b,i),(0,ao,mn-1,0,ret)}^{(MS,MR)}$	$mn \geq 1$
6	$a_{\text{ack_loss_old},(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$	$\text{ack_loss}\langle\text{queue} = [i^{ao}], \text{rn}=i\rangle$	$M_{(2a,i),(mo,ao-1,mn,0,ret)}^{(MS,MR)}$	$ao \geq 1$
7	$a_{\text{receive_dup_ack},(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$	$\text{receive_dup_ack}\langle\text{queue} = [i^{ao-1}], \text{sn}=i, \text{rn}=i\rangle$	$M_{(2a,i),(mo,ao-1,mn,0,ret)}^{(MS,MR)}$	$ao \geq 1$

Table 5.6: The set of arcs, $A_{(2b,i)}^{(MS,MR)}$, with source markings in $V_{(2b,i)}^{(MS,MR)}$.

Row	Arcs	Binding Element	Destination Marking	New Restrictions
1	$a_{\text{timeout_retrans}}^{(MS,MR)}(2b,i),(0,ao,mn,an,ret)$	$\text{timeout_retrans}\langle \text{queue} = [i^{mn}], \text{sn}=i, \text{rc}=ret \rangle$	$M_{(2b,i),(0,ao,mn+1,an,ret+1)}^{(MS,MR)}$	$ret < MR$
2	$a_{\text{mess_loss_new}}^{(MS,MR)}(2b,i),(0,ao,mn,an,ret)$	$\text{mess_loss}\langle \text{queue} = [i^{mn}], \text{sn}=i \rangle$	$M_{(2b,i),(0,ao,mn-1,an,ret)}^{(MS,MR)}$	$mn \geq 1$
3	$a_{\text{receive_mess}}^{(MS,MR)}(2b,i),(0,ao,mn,an,ret)$	$\text{receive_mess}\langle \text{queue} = [i^{mn-1}], \text{sn}=i, \text{rn}=i \oplus_{MS} 1 \rangle$	$M_{(3b,i),(0,ao,mn-1,an,ret)}^{(MS,MR)}$	$mn \geq 1$
4	$a_{\text{ack_loss_old}}^{(MS,MR)}(2b,i),(0,ao,mn,an,ret)$	$\text{ack_loss}\langle \text{queue} = [i^{ao} (i \oplus_{MS} 1)^{an}], \text{rn}=i \rangle$	$M_{(2b,i),(0,ao-1,mn,an,ret)}^{(MS,MR)}$	$ao \geq 1$
5	$a_{\text{ack_loss_new}}^{(MS,MR)}(2b,i),(0,ao,mn,an,ret)$	$\text{ack_loss}\langle \text{queue} = [i^{ao} (i \oplus_{MS} 1)^{an}], \text{rn}=i \oplus_{MS} 1 \rangle$	$M_{(2b,i),(0,ao,mn,an-1,ret)}^{(MS,MR)}$	$an \geq 1$
6	$a_{\text{receive_dup_ack}}^{(MS,MR)}(2b,i),(0,ao,mn,an,ret)$	$\text{receive_dup_ack}\langle \text{queue} = [i^{ao-1} (i \oplus_{MS} 1)^{an}], \text{sn}=i, \text{rn}=i \rangle$	$M_{(2b,i),(0,ao-1,mn,an,ret)}^{(MS,MR)}$	$ao \geq 1$
7	$a_{\text{receive_ack}}^{(MS,MR)}(2b,i),(0,0,mn,an,ret)$	$\text{receive_ack}\langle \text{queue} = [(i \oplus_{MS} 1)^{an-1}], \text{sn}=i, \text{rn}=i \oplus_{MS} 1, \text{rc} = ret \rangle$	$M_{(1,i \oplus_{MS} 1),(mn,an-1,0,0,0)}^{(MS,MR)}$	$an \geq 1$

Table 5.7: The set of arcs, $A_{(3a,i)}^{(MS,MR)}$, with source markings in $V_{(3a,i)}^{(MS,MR)}$, for $MR > 0$.

Row	Arcs	Binding Element	Destination Marking	New Restrictions
1	$a_{\text{timeout_retrans}}^{(MS,MR)}(3a,i),(mo,ao,mn,0,ret)$	$\text{timeout_retrans}\langle \text{queue} = [(i \oplus_{MS} 1)^{mo} i^{mn}], \text{sn}=i, \text{rc}=ret \rangle$	$M_{(3a,i),(mo,ao,mn+1,0,ret+1)}^{(MS,MR)}$	$ret < MR$
2	$a_{\text{mess_loss_old}}^{(MS,MR)}(3a,i),(mo,ao,mn,0,ret)$	$\text{mess_loss}\langle \text{queue} = [(i \oplus_{MS} 1)^{mo} i^{mn}], \text{sn}=i \oplus_{MS} 1 \rangle$	$M_{(3a,i),(mo-1,ao,mn,0,ret)}^{(MS,MR)}$	$mo \geq 1$
3	$a_{\text{mess_loss_new}}^{(MS,MR)}(3a,i),(mo,ao,mn,0,ret)$	$\text{mess_loss}\langle \text{queue} = [(i \oplus_{MS} 1)^{mo} i^{mn}], \text{sn}=i \rangle$	$M_{(3a,i),(mo,ao,mn-1,0,ret)}^{(MS,MR)}$	$mn \geq 1$
4	$a_{\text{ack_loss_old}}^{(MS,MR)}(3a,i),(mo,ao,mn,0,ret)$	$\text{ack_loss}\langle \text{queue} = [i^{ao}], \text{rn}=i \rangle$	$M_{(3a,i),(mo,ao-1,mn,0,ret)}^{(MS,MR)}$	$ao \geq 1$
5	$a_{\text{receive_dup_ack}}^{(MS,MR)}(3a,i),(mo,ao,mn,0,ret)$	$\text{receive_dup_ack}\langle \text{queue} = [i^{ao-1}], \text{sn}=i, \text{rn}=i \rangle$	$M_{(3a,i),(mo,ao-1,mn,0,ret)}^{(MS,MR)}$	$ao \geq 1$
6	$a_{\text{send_ack}}^{(MS,MR)}(3a,i),(mo,ao,mn,0,ret)$	$\text{send_ack}\langle \text{queue} = [i^{ao}], \text{rn}=i \rangle$	$M_{(2a,i),(mo,ao+1,mn,0,ret)}^{(MS,MR)}$	none

Table 5.8: The set of arcs, $A_{(3b,i)}^{(MS,MR)}$, with source markings in $V_{(3b,i)}^{(MS,MR)}$.

Row	Arcs	Binding Element	Destination Marking	New Restrictions
1	$a_{\text{timeout_retrans}}^{(MS,MR)}(3b,i),(0,ao,mn,an,ret)$	$\text{timeout_retrans}\langle \text{queue} = [i^{mn}], \text{sn}=i, \text{rc}=ret \rangle$	$M_{(3b,i),(0,ao,mn+1,an,ret+1)}^{(MS,MR)}$	$ret < MR$
2	$a_{\text{mess_loss_new}}^{(MS,MR)}(3b,i),(0,ao,mn,an,ret)$	$\text{mess_loss}\langle \text{queue} = [i^{mn}], \text{sn}=i \rangle$	$M_{(3b,i),(0,ao,mn-1,an,ret)}^{(MS,MR)}$	$mn \geq 1$
3	$a_{\text{ack_loss_old}}^{(MS,MR)}(3b,i),(0,ao,mn,an,ret)$	$\text{ack_loss}\langle \text{queue} = [i^{ao} (i \oplus_{MS} 1)^{an}], \text{rn}=i \rangle$	$M_{(3b,i),(0,ao-1,mn,an,ret)}^{(MS,MR)}$	$ao \geq 1$
4	$a_{\text{ack_loss_new}}^{(MS,MR)}(3b,i),(0,ao,mn,an,ret)$	$\text{ack_loss}\langle \text{queue} = [i^{ao} (i \oplus_{MS} 1)^{an}], \text{rn}=i \oplus_{MS} 1 \rangle$	$M_{(3b,i),(0,ao,mn,an-1,ret)}^{(MS,MR)}$	$an \geq 1$
5	$a_{\text{receive_dup_ack}}^{(MS,MR)}(3b,i),(0,ao,mn,an,ret)$	$\text{receive_dup_ack}\langle \text{queue} = [i^{ao-1} (i \oplus_{MS} 1)^{an}], \text{sn}=i, \text{rn}=i \rangle$	$M_{(3b,i),(0,ao-1,mn,an,ret)}^{(MS,MR)}$	$ao \geq 1$
6	$a_{\text{receive_ack}}^{(MS,MR)}(3b,i),(0,0,mn,an,ret)$	$\text{receive_ack}\langle \text{queue} = [(i \oplus_{MS} 1)^{an-1}], \text{sn}=i, \text{rn}=i \oplus_{MS} 1, \text{rc} = ret \rangle$	$M_{(4,i \oplus_{MS} 1),(mn,an-1,0,0,0)}^{(MS,MR)}$	$an \geq 1$
7	$a_{\text{send_ack}}^{(MS,MR)}(3b,i),(0,ao,mn,an,ret)$	$\text{send_ack}\langle \text{queue} = [i^{ao} (i \oplus_{MS} 1)^{an}], \text{rn}=i \oplus_{MS} 1 \rangle$	$M_{(2b,i),(0,ao,mn,an+1,ret)}^{(MS,MR)}$	none

Table 5.9: The set of arcs, $A_{(4,i)}^{(MS,MR)}$, with source markings in $V_{(4,i)}^{(MS,MR)}$, for $MR > 0$.

Row	Arcs	Binding Element	Destination Marking	New Restrictions
1	$a_{\text{send_mess}}^{(MS,MR)}(4,i),(mo,ao,0,0,0)$	$\text{send_mess}\langle \text{queue} = [(i \ominus_{MS} 1)^{mo}], \text{sn}=i \rangle$	$M_{(3a,i),(mo,ao,1,0,0)}^{(MS,MR)}$	none
2	$a_{\text{mess_loss_old}}^{(MS,MR)}(4,i),(mo,ao,0,0,0)$	$\text{mess_loss}\langle \text{queue} = [(i \ominus_{MS} 1)^{mo}], \text{sn}=i \ominus_{MS} 1 \rangle$	$M_{(4,i),(mo-1,ao,0,0,0)}^{(MS,MR)}$	$mo \geq 1$
3	$a_{\text{ack_loss_old}}^{(MS,MR)}(4,i),(mo,ao,0,0,0)$	$\text{ack_loss}\langle \text{queue} = [i^{ao}], \text{rn}=i \rangle$	$M_{(4,i),(mo,ao-1,0,0,0)}^{(MS,MR)}$	$ao \geq 1$
4	$a_{\text{receive_dup_ack}}^{(MS,MR)}(4,i),(mo,ao,0,0,0)$	$\text{receive_dup_ack}\langle \text{queue} = [i^{ao-1}], \text{sn}=i, \text{rn}=i \rangle$	$M_{(4,i),(mo,ao-1,0,0,0)}^{(MS,MR)}$	$ao \geq 1$
5	$a_{\text{send_ack}}^{(MS,MR)}(4,i),(mo,ao,0,0,0)$	$\text{send_ack}\langle \text{queue} = [i^{ao}], \text{rn}=i \rangle$	$M_{(1,i),(mo,ao+1,0,0,0)}^{(MS,MR)}$	none

5.6. The Parametric Reachability Graph

and all nodes and arcs are defined in Tables 5.3 to 5.9.

Chapter 6

Parametric Language Analysis

In [35,39] a number of properties were proved correct directly from the algebraic expressions for $RG_{(MS,MR)}$, hence proving them correct for every instance of the infinite class of Stop-and-Wait Protocols. These properties included the size of the RG, absence of unexpected dead markings, absence of livelock, absence of unexpected dead transitions, and the channel bounds.

The next step in the protocol verification methodology [15] is to consider conformance of the SWP to the Stop-and-Wait property of alternating Send and Receive events. The work presented in this chapter is taken from [35].

6.1 The Stop-and-Wait Service Language

The protocol language comprises all sequences of service primitives (user-observable events) exhibited by the protocol. The Stop-and-Wait service is only concerned with the sending and receiving of data. The acknowledgement and retransmission mechanisms (including sequence numbers) are not visible to users of the protocol. Thus the set of service primitives is defined as $SP = \{\text{Send}, \text{Receive}\}$ and the Stop-and-Wait Service language is defined as:

Definition 33 (Service Language).

The Stop-and-Wait Service Language, \mathcal{L}_S , of alternating send and receive events is given by the regular expression $(\text{Send Receive})^ \text{Send}^\dagger$ where Send^\dagger represents 0 or 1 repetitions of the **Send** primitive.*

This service language is illustrated using a Finite State Automaton (FSA) in Fig. 6.1. The service language specifies sequences of alternating send and receive events, which may end with a **Send** or a **Receive** event. Because the service should specify more than just sequences of events, the Stop-and-Wait service language is sometimes referred to as the *Stop-and-Wait Property* of alternating send and receive events. When the Stop-and-Wait protocol is operating correctly, one message will be received for every original message sent. It may seem unusual at first glance that sequences ending in **Send** are allowed by the service. This indicates that the last **Send** in a sequence may not be followed by a corresponding **Receive** if the last message that was sent and all retransmissions are lost. This corresponds to the situation where the sender gives up trying to get the message to the receiver on the basis that the link is down. Dealing with this situation is the job of a management entity which is not part of the Stop-and-Wait Protocol, and so is not reflected in the service language - the sender simply stops. Were the underlying medium lossless, the service could well be specified as $(\text{Send Receive})^*$, as the expected behaviour would not include the possibility of loss of a message and all retransmissions. This may also be the case over a lossy medium if unbounded retransmission was considered, given suitable fairness assumptions about loss in the channel. Neither of the last two situations are considered in this report.

In this chapter the protocol language of the SWP is derived from the parametric RG of Theorem 1, allowing the protocol language of the infinite class of SWPs to be represented parametrically. Obtaining such a parametric expression for the protocol language enables the conformance of the SWP to its service of alternating send

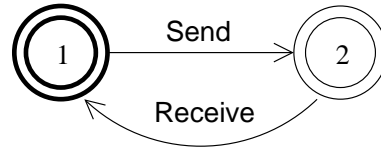


Figure 6.1: An FSA of the Stop-and-Wait Service Language.

and receive events to be verified for *all* values of the `MaxSeqNo` and `MaxRetrans` parameters. We state the property of conformance of the SWP to its service language in the following theorem.

Theorem 2. *The Stop-and-Wait protocol, as defined by the CPN in Figs. 4.1 and 4.2 conforms to the Stop-and-Wait service of alternating send and receive events, i.e. $\mathcal{L}_P = \mathcal{L}_S = (\mathbf{Send}, \mathbf{Receive})^* \mathbf{Send}^\dagger$, for all values of the `MaxSeqNo` and `MaxRetrans` parameters.*

This theorem is proved progressively throughout this chapter, and is structured as follows. We begin in Section 6.2 by obtaining a parametric Finite State Automaton from $RG_{(MS,MR)}$ that represents our protocol language. We then calculate a parametric ϵ -closure of selected parametric markings in Section 6.3, for use in the parametric determinisation procedure combining ϵ -removal with lazy subset evaluation, which is undertaken in Section 6.4. Finally, in Section 6.5, we perform a minimisation procedure on the deterministic version of our parametric FSA and compare the result with the Stop-and-Wait service of alternating send and receive events.

6.2 Obtaining a Parametric Representation of the Protocol Language

By interpreting the RG as a Finite State Automaton (FSA) and relabelling binding elements as either service primitives or epsilon (empty) moves, standard algorithms [8,63] can be used to obtain the minimal deterministic FSA of the protocol language.

In accordance with the methodology presented in [15] we begin by defining a mapping from the binding elements of $CPN_{(MS,MR)}$ to service primitives or ϵ :

Definition 34 (Mapping from Binding Elements to Service Primitives).

Let $Prim : BE_{(MS,MR)} \rightarrow SP \cup \{\epsilon\}$ be a mapping from the set of binding elements of $CPN_{(MS,MR)}$ to either a service primitive name or to ϵ , where

- $BE_{(MS,MR)}$ is the set of binding elements that occur in $CPN_{(MS,MR)}$; and
- $SP = \{\mathbf{Send}, \mathbf{Receive}\}$;

such that, for $0 \leq i \leq MS$, $0 \leq x \leq MR$, and $(t, b) \in BE_{(MS,MR)}$:

$$Prim((t, b)) = \begin{cases} \mathbf{Send}, & \text{if } (t, b) = \mathbf{send_mess} \langle \text{queue} = [(i \ominus_{MS} 1)^x], sn = i \rangle, \\ \mathbf{Receive}, & \text{if } (t, b) = \mathbf{receive_mess} \langle \text{queue} = [i^x], sn = i, rn = i \rangle, \\ \epsilon, & \text{otherwise.} \end{cases}$$

The mapping $Prim$ will map all occurrences of the `send_mess` transition to the primitive `Send` and only those occurrences of `receive_mess` corresponding to acceptance of a new message ($sn = rn$) to the primitive `Receive`. All other binding elements (including occurrences of `receive_mess` corresponding to detection and discarding of a duplicate) are mapped to ϵ .

All that remains to interpret $RG_{(MS,MR)}$ as a FSA is to define the initial and halt states. We define the initial state of the FSA as the initial marking of $CPN_{(MS,MR)}$, i.e. $M_0 = M_{(1,0),(0,0,0,0)}^{(MS,MR)}$. We have an arbitrary

number of messages to send from the sender to the receiver, and so we define a legitimate halt state as any state in which $l \in \mathbb{N}$ messages have been transmitted and successfully acknowledged, so that both the sender and receiver are in their ready states and there are no messages or acknowledgements in the channel. This corresponds to the markings $M_{(1,i),(0,0,0,0,0)}^{(MS,MR)}$ for all $0 \leq i \leq MS$ (incorporating the initial marking). We also include the dead markings of $RG_{(MS,MR)}$ in the set of halt states, i.e. $M_{(2a,i),(0,0,0,0,MR)}^{(MS,MR)}$ and $M_{(2b,i),(0,0,0,0,MR)}^{(MS,MR)}$ for all $0 \leq i \leq MS$ (this set of dead markings is proved correct in [35]). In line with our definition of the SWP service, these dead markings represent expected halt states of the protocol when operating over a lossy medium.

We are now ready to define the FSA associated with $RG_{(MS,MR)}$:

Definition 35 ($FSA_{RG_{(MS,MR)}}$).

The FSA associated with $RG_{(MS,MR)} = (V_{(MS,MR)}, A_{(MS,MR)})$ of $CPN_{(MS,MR)}$, with initial marking M_0 , is $FSA_{RG_{(MS,MR)}} = (V_{(MS,MR)}, SP, \Delta_{(MS,MR)}, M_0, F_{(MS,MR)})$ where

- $SP = \{\text{Send, Receive}\}$ is the set of service primitive names of interest (the alphabet of the FSA);
- $\Delta_{(MS,MR)} = \{M, Prim((t, b), M') \mid (M, (t, b), M') \in A_{(MS,MR)}\}$ is the set of transitions labelled by service primitives or epsilons for internal events (the transition relation of the FSA); and
- $F_{(MS,MR)} = \{M_{(1,i),(0,0,0,0,0)}^{(MS,MR)}, M_{(2a,i),(0,0,0,0,MR)}^{(MS,MR)}, M_{(2b,i),(0,0,0,0,MR)}^{(MS,MR)} \mid 0 \leq i \leq MS\}$ is the set of final states.

The states of $FSA_{RG_{(MS,MR)}}$ are the nodes of $RG_{(MS,MR)}$, given by Table 5.3. The arcs of $FSA_{RG_{(MS,MR)}}$ are given by Tables 5.4 to 5.9 by applying $Prim$ to each binding element (arc label). The arcs defined by row 1 of Table 5.4 and row 1 of Table 5.9 correspond to binding elements that map to the primitive **Send**. The arcs defined by row 5 of Table 5.5 correspond to binding elements that map to the primitive **Receive**. All other arcs have binding elements that map to ϵ .

In the remainder of this chapter, we will often refer to edges in $FSA_{RG_{(MS,MR)}}$ by the corresponding arc in $RG_{(MS,MR)}$. In particular, we will refer to the transition or binding element that labels the corresponding arc in $RG_{(MS,MR)}$. This provides a way of easily identifying the edges whose labels are mapped to ϵ by $Prim$.

6.3 Epsilon Closures

Before proceeding with FSA reduction we calculate symbolically the ϵ -closure of all markings in $V_{(2a,i)}^{(MS,MR)}$ and $V_{(3b,i)}^{(MS,MR)}$ of Table 5.3. As will become evident, we do not need to determine symbolic expressions for the ϵ -closure of markings in $V_{(1,i)}^{(MS,MR)}$, $V_{(2b,i)}^{(MS,MR)}$, $V_{(3a,i)}^{(MS,MR)}$ or $V_{(4,i)}^{(MS,MR)}$. This is due to the lazy subset evaluation technique for determinisation [41] used in Section 6.4.

To calculate the ϵ -closure of a marking, M , we use the following method. The first step is to explore the RG along arcs whose labels map to ϵ via $Prim$ to create a spanning of markings reachable by one or more ϵ moves. This creates a subgraph of $FSA_{RG_{(MS,MR)}}$, whose edges are ϵ moves and M is the origin. The second step is to show that all markings reachable by 0 or more ϵ moves are actually covered by the subgraph we created, so that the set of nodes spanned by this subgraph is actually the ϵ -closure of M .

As a final step, we determine the set of outgoing edges from the markings in the ϵ -closure that are not labelled by ϵ , i.e. are labelled by service primitives. These sets of edges follow directly from the proofs of the ϵ -closures and the relevant arc tables, and are thus correct by construction. These are used in the determinisation procedure in Section 6.4.

6.3.1 The ϵ -closure of Nodes in $V_{(2a,i)}^{(MS,MR)}$

Recall from row 2 of Table 5.3 that $V_{(2a,i)}^{(MS,MR)} = \{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq ret \leq MR, 0 \leq mn \leq ret + 1\}$. Table 5.5 defines all outgoing arcs from markings in $V_{(2a,i)}^{(MS,MR)}$. Row 5 defines

arcs whose action maps to the service primitive **Receive**. The remaining rows define arcs whose actions map to ϵ . Note that although rows 4 and 5 both correspond to an occurrence of the `receive_mess` transition, row 4 corresponds to reception of an old duplicate and thus maps to an ϵ move, unlike the arcs defined by row 5.

We now begin exploring from a marking, $M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \in V_{(2a,i)}^{(MS,MR)}$. From the enabling rules of CPNs, transition `receive_mess` (row 4 of Table 5.5) is enabled by any marking $M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$ if $mo > 0$. This leads to a marking $M_{(3a,i),(mo-1,ao,mn,0,ret)}^{(MS,MR)} \in V_{(3a,i)}^{(MS,MR)}$. From this marking, `send_ack` (row 6 of Table 5.7) is always enabled, leading to a marking $M_{(2a,i),(mo-1,ao+1,mn,0,ret)}^{(MS,MR)} \in V_{(2a,i)}^{(MS,MR)}$. The net result is one fewer old message in the channel, and one more old acknowledgement. This repeated transition sequence can occur mo number of times, i.e. until there are no more old messages in the channel. This generates two sets of markings, V_a (class 2a markings) and V_b (class 3a markings), the elements of which are all reachable from $M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$ by the occurrence of 0 or more ϵ transitions:

$$\begin{aligned} V_a &= \{M_{(2a,i),(mo-x,ao+x,mn,0,ret)}^{(MS,MR)} \mid 0 \leq x \leq mo\} \\ &= \{M_{(2a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \mid mo' = mo - x, ao' = ao + x, 0 \leq x \leq mo\} \end{aligned} \quad (6.1)$$

and

$$\begin{aligned} V_b &= \{M_{(3a,i),(mo-1-x,ao+x,mn,0,ret)}^{(MS,MR)} \mid 0 \leq x \leq mo - 1, mo > 0\} \\ &= \{M_{(3a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \mid mo' = mo - 1 - x, ao' = ao + x, 0 \leq x \leq mo - 1, mo > 0\} \end{aligned} \quad (6.2)$$

Again, technically, V_a and V_b define parameterised families of sets, one instance of each for each marking in $V_{(2a,i)}^{(MS,MR)}$ for each allowable combination of parameter values. However, we omit this detail from the name of the sets for notational simplicity.

Equation (6.1) can be simplified by eliminating x , provided the relationship between mo' and ao' induced by x is preserved. Summing the expressions for mo' and ao' gives $mo' + ao' = mo + ao$, which captures the relationship between mo' and ao' induced by x . Given this, x can be eliminated from the expressions for mo' and ao' by observing that as x varies from 0 to mo , the value of mo' varies from mo to 0 and the value of ao' varies from ao to $ao + mo$. This is captured by the expressions $0 \leq mo' \leq mo$ and $ao \leq ao' \leq mo + ao$. However, only one of these is required to fully define the values of mo' and ao' , because of the expression $mo' + ao' = mo + ao$. The variable x can be eliminated from Equation (6.2) in a similar way, resulting in the following expressions for V_a and V_b :

$$V_a = \{M_{(2a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \mid mo' + ao' = mo + ao, 0 \leq mo' \leq mo\} \quad (6.3)$$

and

$$V_b = \{M_{(3a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \mid mo' + ao' = mo + ao - 1, 0 \leq mo' \leq mo - 1, mo > 0\} \quad (6.4)$$

The sets V_a and V_b are illustrated in the context of $V_{(2a,i)}^{(MS,MR)}$ and $V_{(3a,i)}^{(MS,MR)}$ by Fig. 6.2. On the left is $V_{(2a,i)}^{(MS,MR)}$ and on the right is $V_{(3a,i)}^{(MS,MR)}$. The nodes explicitly shown within $V_{(2a,i)}^{(MS,MR)}$ constitute V_a and the nodes explicitly shown within $V_{(3a,i)}^{(MS,MR)}$ constitute V_b . The relationship between the class 2a and class 3a markings, by the repeated firing of `receive_mess` followed by `send_ack` used to generate V_a and V_b is illustrated by the arcs going forwards and backwards from $V_{(2a,i)}^{(MS,MR)}$ to $V_{(3a,i)}^{(MS,MR)}$ in the centre of the diagram. As can be seen in this diagram, every time `receive_mess` occurs, the number of old messages decreases by one, and every time `send_ack` occurs, the number of old acknowledgements increases by one.

The transition `timeout_retrans` is enabled by those markings in V_a and V_b in which the maximum number of retransmissions has not yet been reached, i.e. $ret < MR$ (row 1 of Table 5.5 and Table 5.7 respectively).

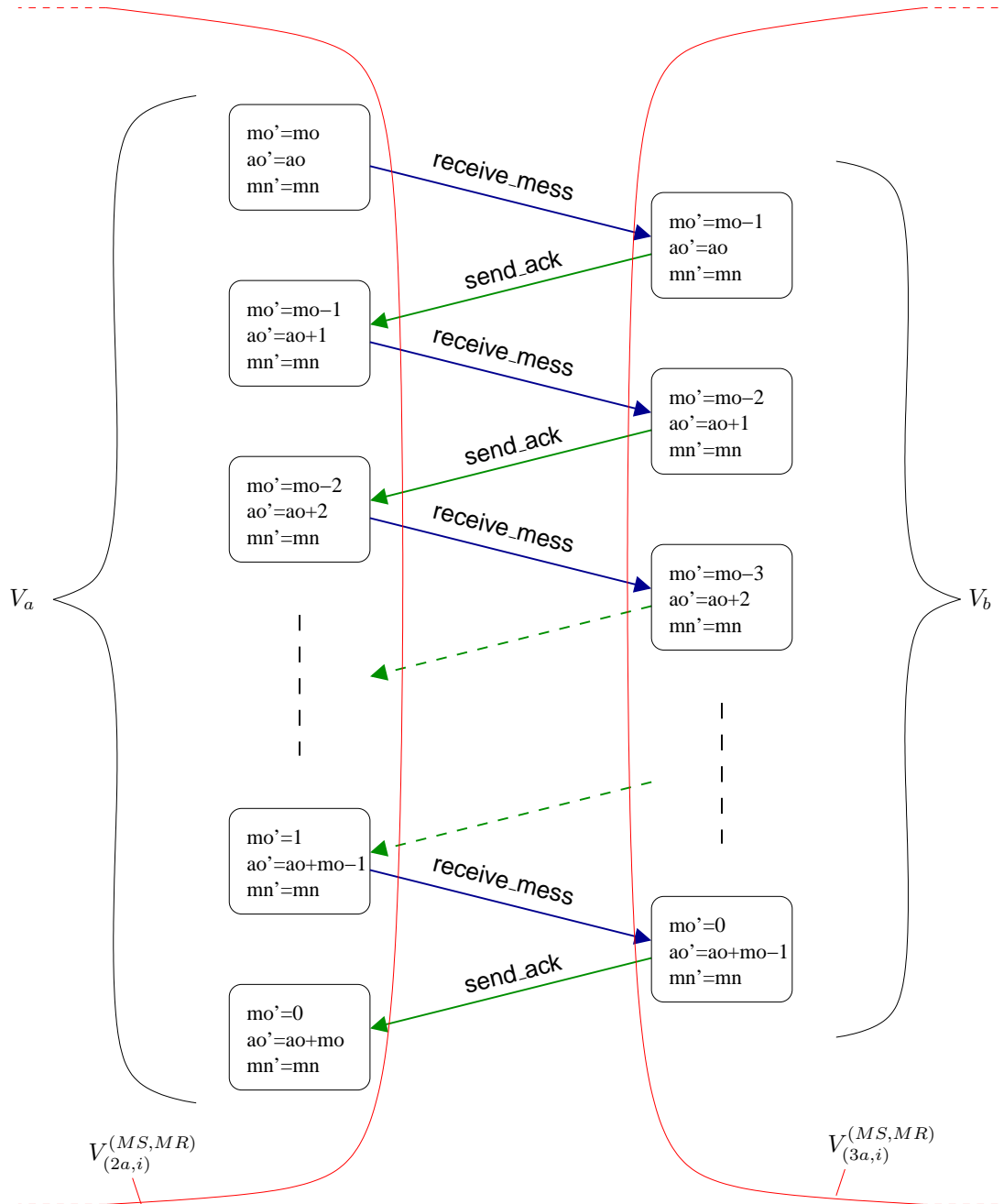


Figure 6.2: The markings, represented by mo , ao and mn , reachable from $M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$ through repeated successive firings of *receive_mess* followed by *send_ack*.

This transition is enabled and can occur up to $MR - ret$ number of times consecutively from each such marking, i.e. until the maximum number of retransmissions is reached. From Row 1 of both Table 5.5 and Table 5.7, firing `timeout_retrans` from each marking in V_a and V_b results in the following sets of markings reachable by 0 or more ϵ moves:

$$V'_a = \{M_{(2a,i),(mo',ao',mn+y,0,ret+y)}^{(MS,MR)} \mid M_{(2a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \in V_a, 0 \leq y \leq MR - ret\} \quad (6.5)$$

$$V'_b = \{M_{(3a,i),(mo',ao',mn+y,0,ret+y)}^{(MS,MR)} \mid M_{(3a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \in V_b, 0 \leq y \leq MR - ret\} \quad (6.6)$$

Substituting (6.3) and (6.4) into (6.5) and (6.6) respectively gives:

$$V'_a = \{M_{(2a,i),(mo',ao',mn',0,ret')}^{(MS,MR)} \mid mo' + ao' = mo + ao, 0 \leq mo' \leq mo, mn' = mn + y, \\ ret' = ret + y, 0 \leq y \leq MR - ret\} \quad (6.7)$$

$$V'_b = \{M_{(3a,i),(mo',ao',mn',0,ret')}^{(MS,MR)} \mid mo' + ao' = mo + ao - 1, 0 \leq mo' \leq mo - 1, mn' = mn + y, \\ ret' = ret + y, 0 \leq y \leq MR - ret, mo > 0\} \quad (6.8)$$

Substituting the expression $y = ret' - ret$ into (6.7) and (6.8) to eliminate y gives:

$$V'_a = \{M_{(2a,i),(mo',ao',mn',0,ret')}^{(MS,MR)} \mid mo' + ao' = mo + ao, 0 \leq mo' \leq mo, mn' = mn + ret' - ret, \\ ret \leq ret' \leq MR\} \quad (6.9)$$

$$V'_b = \{M_{(3a,i),(mo',ao',mn',0,ret')}^{(MS,MR)} \mid mo' + ao' = mo + ao - 1, 0 \leq mo' \leq mo - 1, \\ mn' = mn + ret' - ret, ret \leq ret' \leq MR, mo > 0\} \quad (6.10)$$

The sets V'_a and V'_b are illustrated in the context of $V_{(2a,i)}^{(MS,MR)}$ and $V_{(3a,i)}^{(MS,MR)}$ by Fig. 6.3. From each marking identified in V_a and V_b , as illustrated in Fig. 6.2, the `timeout_retrans` transition can occur repeatedly until the maximum number of retransmissions is reached. This is shown by the additional markings branching out from the left of those from V_a and from the right of those from V_b . As can be seen in this diagram, the value of mn' increases for each occurrence of `timeout_retrans`, until it reaches its maximum possible value of $mn' = mn + MR - ret$, at which point it is no longer enabled.

Transitions `mess_loss` (rows 2 and 3 of Table 5.5) and `ack_loss` (row 6 of Table 5.5) allow all markings in the downward-closed set of each marking in V'_a to be reached. Similarly, rows 2, 3 and 4 of Table 5.7 allow the same for V'_b . Applying Definition 30 to (6.9) and (6.10) to obtain the downward-closed sets of all markings in V'_a and V'_b gives:

$$V''_a = DC(V'_a) \\ = \{M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \mid M_{(2a,i),(mo',ao',mn',0,ret')}^{(MS,MR)} \in V'_a, mo'' \leq mo', ao'' \leq ao', mn'' \leq mn'\} \\ = \{M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \mid mo'' \leq mo', ao'' \leq ao', mn'' \leq mn', mo' + ao' = mo + ao, \\ 0 \leq mo' \leq mo, mn' = mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.11)$$

$$V''_b = DC(V'_b) \\ = \{M_{(3a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \mid M_{(3a,i),(mo',ao',mn',0,ret')}^{(MS,MR)} \in V'_b, mo'' \leq mo', ao'' \leq ao', mn'' \leq mn'\} \\ = \{M_{(3a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \mid mo'' \leq mo', ao'' \leq ao', mn'' \leq mn', mo' + ao' = mo + ao - 1, \\ 0 \leq mo' \leq mo - 1, mn' = mn + ret' - ret, ret \leq ret' \leq MR, mo > 0\} \quad (6.12)$$

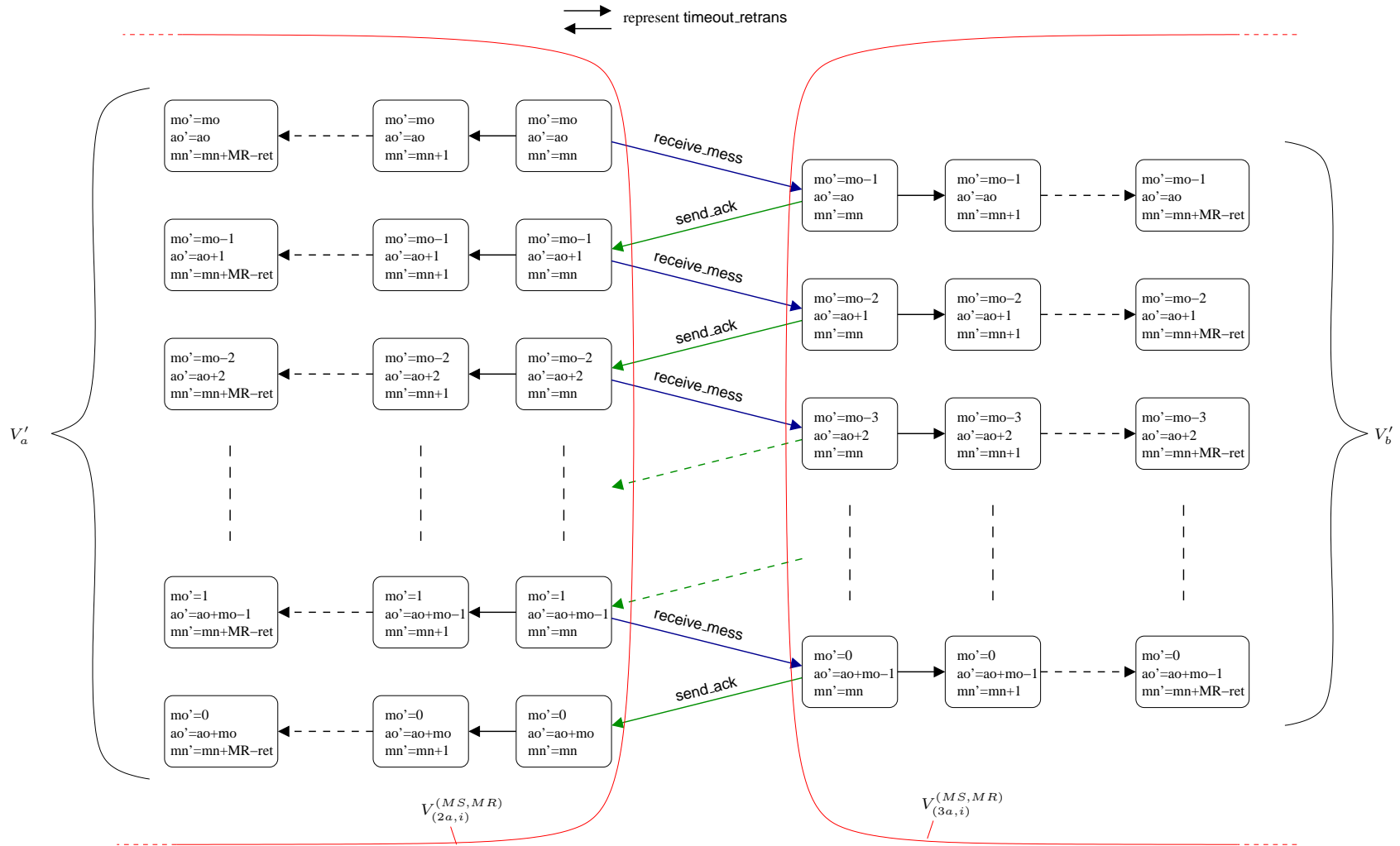


Figure 6.3: The markings from $V_{(2a,i)}^{(MS,MR)}$ and $V_{(3a,i)}^{(MS,MR)}$, represented by mo , ao and mn , reachable from the markings in Fig. 6.2 through repeated firing of `timeout_retrans`.

The variables mo' , ao' and mn' can be eliminated in (6.11) and (6.12) by substituting the inequalities $mo'' \leq mo'$, $ao'' \leq ao'$ and $mn'' \leq mn'$ wherever mo' , ao' and mn' appear, giving:

$$V_a'' = \{M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \mid mo'' + ao'' \leq mo + ao, 0 \leq mo'' \leq mo, mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.13)$$

$$V_b'' = \{M_{(3a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \mid mo'' + ao'' \leq mo + ao - 1, 0 \leq mo'' \leq mo - 1, mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR, mo > 0\} \quad (6.14)$$

We do not attempt to visualise V_a'' or V_b'' in the same level of detail as V_a , V_b , V_a' and V_b' in Figs. 6.2 and 6.3, as it becomes a complex three-dimensional lattice of markings. Instead, we take a more abstract view and illustrate the coverage of V_a'' and V_b'' over $V_i^{(MS,MR)}$ in Fig. 6.4. The shaded region in $V_{(2a,i)}^{(MS,MR)}$ represents V_a'' and the shaded region in $V_{(3a,i)}^{(MS,MR)}$ represents V_b'' . Naturally, the coverage will depend on which specific initial marking, $M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$ (the large red circle), is chosen. For example, if $M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)}$ is chosen as the initial marking, then all of $V_{(2a,i)}^{(MS,MR)}$ is covered by V_a'' .

At this point we conjecture that $V_a'' \cup V_b''$ is the ϵ -closure of a marking $M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}$. We must ensure that there are no additional markings reachable via ϵ moves:

Lemma 4. Any marking $M' \in V_{(MS,MR)}$ reachable via an ϵ move from a marking $M \in V_a'' \cup V_b''$ is also contained in $V_a'' \cup V_b''$, i.e. $\forall M \in V_a'' \cup V_b'', M \xrightarrow{\epsilon} M' \implies M' \in V_a'' \cup V_b''$.

Proof. For $M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_a''$ at most 6 outgoing ϵ moves are defined by Table 5.5 and mapping *Prim*. They are treated systematically below. The following relies heavily on Equations (6.13) and (6.14) and references are made when appropriate.

mess_loss_old (row 2) is enabled by $M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_a''$ if $mo'' > 0$. Its occurrence results in a marking $M_{(2a,i),(mo''-1,ao'',mn'',0,ret')}^{(MS,MR)}$.

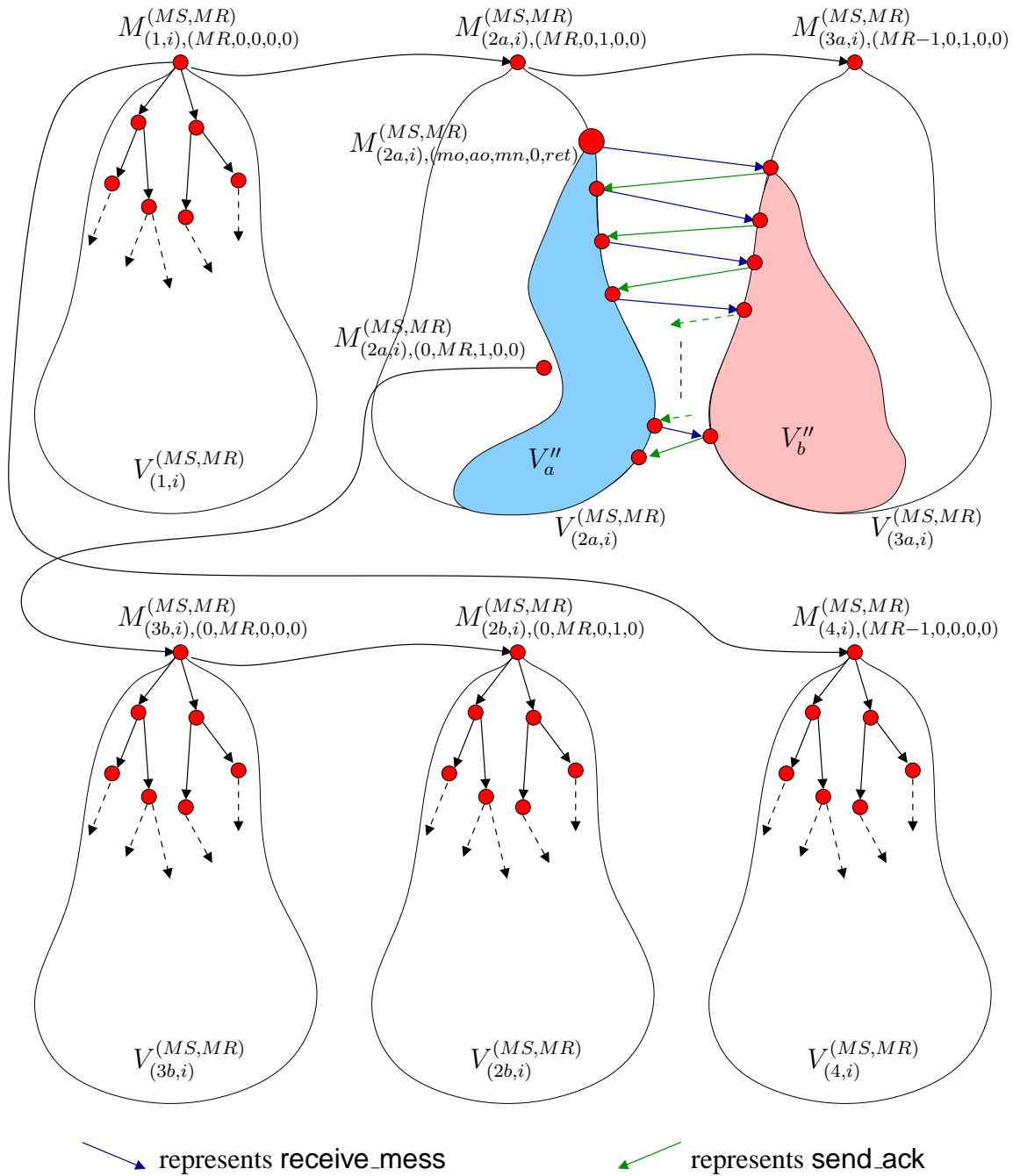
mess_loss_new (row 3) is enabled by $M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_a''$ if $mn'' > 0$. Its occurrence results in a marking $M_{(2a,i),(mo'',ao'',mn''-1,0,ret')}^{(MS,MR)}$.

ack_loss_old, receive_dup_ack (rows 6 and 7) are enabled by $M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_a''$ if $ao'' > 0$. Its occurrence of either results in a marking $M_{(2a,i),(mo'',ao''-1,mn'',0,ret')}^{(MS,MR)}$.

All three of these resulting markings are identical to their source marking, $M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_a''$, except for either one less old message, one less new message, or one less old acknowledgement. Because V_a'' is a downward-closed set, and the actions resulting in these three markings correspond to loss, then by the definition of a downward closed set (Definition 30) all three resulting markings are also elements of the downward-closed set, V_a'' .

The remaining two outgoing ϵ moves require more careful treatment:

timeout_retrans (row 1) is enabled by $M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_a''$ if $ret' < MR$. Its occurrence results in a marking $M_{(2a,i),(mo'',ao'',mn''+1,0,ret'+1)}^{(MS,MR)}$. From the inequalities in Equation (6.13), $(mn'' + 1) \leq mn + (ret' + 1) - ret$, and $ret < (ret' + 1) \leq MR$ because of the restriction that $ret' < MR$. These do not violate the inequalities given in Equation (6.13) when substituting $(mn'' + 1)$ and $(ret' + 1)$ for mn'' and ret' respectively into Equation (6.13). Hence $M_{(2a,i),(mo'',ao'',mn''+1,0,ret'+1)}^{(MS,MR)} \in V_a''$.


 Figure 6.4: The coverage of V_a'' and V_b'' over $V_i^{(MS,MR)}$.

receive_mess (row 4) is enabled by $M_{(2a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_a''$ if $mo'' > 0$. Its occurrence results in a marking $M_{(3a,i),(mo''-1,ao'',mn'',0,ret')}^{(MS,MR)}$. From the inequalities in Equation (6.13), we can determine that $(mo'' - 1) + ao'' \leq mo + ao - 1$, and $0 \leq (mo'' - 1) \leq mo - 1$ because of the restriction that $mo'' > 0$. These match the corresponding inequalities in Equation (6.14) when substituting $(mo'' - 1)$ for mo'' in Equation (6.14). By inspection, the other inequalities from Equation (6.13) are the same as corresponding inequalities in Equation (6.14). Hence $M_{(3a,i),(mo''-1,ao'',mn'',0,ret')}^{(MS,MR)} \in V_b''$.

For $M_{(3a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_b''$ at most 6 outgoing ϵ moves are defined by Table 5.7 and mapping

Prim. They are treated systematically below. Again, the following relies heavily on Equations (6.13) and (6.14) and references are made when appropriate.

mess_loss_old (row 2) is enabled by $M_{(3a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_b''$ if $mo'' > 0$. Its occurrence leads to a marking $M_{(3a,i),(mo''-1,ao'',mn'',0,ret')}^{(MS,MR)}$.

mess_loss_new (row 3) is enabled by $M_{(3a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_b''$ if $mn'' > 0$. Its occurrence leads to a marking $M_{(3a,i),(mo'',ao'',mn''-1,0,ret')}^{(MS,MR)}$.

ack_loss_old, receive_dup_ack (rows 4 and 5) are enabled by $M_{(3a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_b''$ if $ao'' > 0$. Its occurrence of either leads to a marking $M_{(3a,i),(mo'',ao''-1,mn'',0,ret')}^{(MS,MR)}$.

All three of these resulting markings are identical to their source marking, $M_{(3a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_b''$, except for either one less old message, one less new message, or one less old acknowledgement. As was seen previously, because V_b'' is a downward-closed set, and the actions leading to these three markings all correspond to loss (receive_dup_ack has the same effect as ack_loss_old), then by the definition of a downward closed set (Definition 30), all three resulting markings are also elements of V_b'' .

The remaining two outgoing ϵ moves require more careful treatment:

timeout_retrans (row 1) is enabled by $M_{(3a,i),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \in V_b''$ if $ret' < MR$. Its occurrence results in a marking $M_{(3a,i),(mo'',ao'',mn''+1,0,ret'+1)}^{(MS,MR)}$. From the inequalities in Equation (6.14), $(mn'' + 1) \leq mn + (ret' + 1) - ret$, and $ret < (ret' + 1) \leq MR$ because of the restriction that $ret' < MR$. These do not violate the inequalities given in Equation (6.14) when substituting $(mn'' + 1)$ and $(ret' + 1)$ for mn'' and ret' into Equation (6.14). Hence $M_{(3a,i),(mo'',ao'',mn''+1,0,ret'+1)}^{(MS,MR)} \in V_b''$.

send_ack (row 6) is enabled by all markings in V_b'' . Its occurrence leads to a marking $M_{(2a,i),(mo'',ao''+1,mn'',0,ret')}^{(MS,MR)}$. From the first inequality in Equation (6.14), $mo'' + (ao'' + 1) \leq mo + ao$. This matches the corresponding inequality in Equation (6.13) when substituting $(ao'' + 1)$ for ao'' in Equation (6.13). By inspection, the other inequalities from Equation (6.14) are the same as corresponding inequalities in Equation (6.13). Hence $M_{(2a,i),(mo'',ao''+1,mn'',0,ret')}^{(MS,MR)} \in V_a''$.

All successors of all markings in $V_a'' \cup V_b''$ reachable by an ϵ move have been explored, and all are contained in $V_a'' \cup V_b''$, i.e. $\{M' \mid M \xrightarrow{\epsilon} M', M \in V_a'' \cup V_b''\} \subseteq V_a'' \cup V_b''$. Thus the lemma is proved. \square

Corollary 1. *The ϵ -closure of a marking $M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \in V_{(2a,i)}^{(MS,MR)}$, $Closure(M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)})$, is given by the union of the parameterised sets V_a'' and V_b'' defined by Equations (6.13) and (6.14) respectively, i.e. (replacing double primes with single primes):*

$$\begin{aligned} Closure(M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}) &= \{M_{(2a,i),(mo',ao',mn',0,ret')}^{(MS,MR)} \mid mo' + ao' \leq mo + ao, \\ &\quad 0 \leq mo' \leq mo, mn' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \\ &\cup \{M_{(3a,i),(mo',ao',mn',0,ret')}^{(MS,MR)} \mid mo' + ao' \leq mo + ao - 1, 0 \leq mo' \leq mo - 1, \\ &\quad mn' \leq mn + ret' - ret, ret \leq ret' \leq MR, mo > 0\} \quad (6.15) \end{aligned}$$

The only transition enabled by the markings in $Closure(M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)})$ that does not map to ϵ is given by Row 5 of Table 5.5 and maps to the Receive service primitive. From Row 5 of Table 5.5, the

corresponding set of non- ϵ -labelled outgoing edges from all markings in the above closure is given by:

$$\begin{aligned} \text{OutEdges}(\text{Closure}(M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)})) &= \{(M_{(2a,i),(0,ao',mn',0,ret')}^{(MS,MR)}, \text{Receive}, \\ &M_{(3b,i),(0,ao',mn'-1,0,ret')}^{(MS,MR)} \mid M_{(2a,i),(0,ao',mn',0,ret')}^{(MS,MR)} \in V_a'', mn' > 0\} \end{aligned} \quad (6.16)$$

Substituting V_a'' from Equation (6.13) into Equation (6.16) we obtain:

$$\begin{aligned} \text{OutEdges}(\text{Closure}(M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)})) &= \{(M_{(2a,i),(0,ao',mn',0,ret')}^{(MS,MR)}, \text{Receive}, \\ &M_{(3b,i),(0,ao',mn'-1,0,ret')}^{(MS,MR)} \mid ao' \leq mo + ao, mn' \leq mn + ret' - ret, ret \leq ret' \leq MR, mn' > 0\} \end{aligned} \quad (6.17)$$

6.3.2 The ϵ -closure of Nodes in $V_{(3b,i)}^{(MS,MR)}$

Recall from row 5 of Table 5.3 that $V_{(3b,i)}^{(MS,MR)} = \{M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq ret \leq MR, 0 \leq mn + an \leq ret\}$. Table 5.8 defines all outgoing arcs from markings in $V_{(3b,i)}^{(MS,MR)}$. All rows define arcs whose action maps to ϵ .

The transition `timeout_retrans` (row 1 of Table 5.8) is enabled by all markings $M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \in V_{(3b,i)}^{(MS,MR)}$ in which $ret < MR$. Its occurrence leads to a marking $M_{(3b,i),(0,ao,mn+1,an,ret+1)}^{(MS,MR)}$. The net result is one extra new message in the channel and a retransmission counter incremented by 1. Transition `timeout_retrans` is enabled, and can occur, up to $MR - ret$ number of times consecutively from $M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)}$, i.e. until the maximum number of retransmissions is reached. The occurrence of `timeout_retrans` from $M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \in V_{(3b,i)}^{(MS,MR)}$, 0 to $MR - ret$ number of times, results in the set of markings, V_c , the elements of which are reachable from $M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)}$ by 0 or more ϵ moves:

$$\begin{aligned} V_c &= \{M_{(3b,i),(0,ao,mn+x,an,ret+x)}^{(MS,MR)} \mid 0 \leq x \leq MR - ret\} \\ &= \{M_{(3b,i),(0,ao,mn',an,ret')}^{(MS,MR)} \mid mn' = mn + x, ret' = ret + x, 0 \leq x \leq MR - ret\} \end{aligned} \quad (6.18)$$

Equation (6.18) can be simplified by eliminating x , by substituting $x = ret' - ret$:

$$V_c = \{M_{(3b,i),(0,ao,mn',an,ret')}^{(MS,MR)} \mid mn' = mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.19)$$

Transition `send_ack` (row 7 of Table 5.8) is enabled by any marking $M_{(3b,i),(0,ao,mn',an,ret')}^{(MS,MR)} \in V_c$. This leads to a marking $M_{(2b,i),(0,ao,mn',an+1,ret')}^{(MS,MR)} \in V_{(2b,i)}^{(MS,MR)}$. From this marking, `receive_mess` (row 3 of Table 5.6) is enabled, and can occur, provided $mn' > 0$, leading to a marking $M_{(3b,i),(0,ao,mn'-1,an+1,ret')}^{(MS,MR)}$. This sequence of transitions can occur up to mn' number of times, finishing with an occurrence of `send_ack`, i.e. until there are no more new messages in the channel. This generates two sets of markings, V_d (class 3b markings) and V_e (class 2b markings), the elements of which are all reachable from $M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)}$ by 0 or more ϵ transitions:

$$V_d = \{M_{(3b,i),(0,ao,mn'-y,an+y,ret')}^{(MS,MR)} \mid M_{(3b,i),(0,ao,mn',an,ret')}^{(MS,MR)} \in V_c, 0 \leq y \leq mn'\} \quad (6.20)$$

$$V_e = \{M_{(2b,i),(0,ao,mn'-y,an+1+y,ret')}^{(MS,MR)} \mid M_{(3b,i),(0,ao,mn',an,ret')}^{(MS,MR)} \in V_c, 0 \leq y \leq mn'\} \quad (6.21)$$

Using Equation (6.19) in Equations (6.20) and (6.21) gives:

$$\begin{aligned}
 V_d &= \{M_{(3b,i),(0,ao,mn'',an',ret')}^{(MS,MR)} \mid mn'' = mn' - y, an' = an + y, mn' = mn + ret' - ret, \\
 &\hspace{15em} ret \leq ret' \leq MR, 0 \leq y \leq mn'\} \\
 &= \{M_{(3b,i),(0,ao,mn'',an',ret')}^{(MS,MR)} \mid mn'' = mn + ret' - ret - y, an' = an + y, ret \leq ret' \leq MR, \\
 &\hspace{15em} 0 \leq y \leq mn + ret' - ret\} \quad (6.22)
 \end{aligned}$$

$$\begin{aligned}
 V_e &= \{M_{(2b,i),(0,ao,mn'',an',ret')}^{(MS,MR)} \mid mn'' = mn' - y, an' = an + 1 + y, mn' = mn + ret' - ret, \\
 &\hspace{15em} ret \leq ret' \leq MR, 0 \leq y \leq mn'\} \\
 &= \{M_{(2b,i),(0,ao,mn'',an',ret')}^{(MS,MR)} \mid mn'' = mn + ret' - ret - y, an' = an + 1 + y, ret \leq ret' \leq MR, \\
 &\hspace{15em} 0 \leq y \leq mn + ret' - ret\} \quad (6.23)
 \end{aligned}$$

Equation (6.22) can be simplified by eliminating y , provided the relationship between mn'' and an' induced by y is preserved. Summing the expressions for mn'' and an' gives $mn'' + an' = mn + an + ret' - ret$, which captures the relationship between mn'' and an' induced by y . Given this, y can be eliminated from the expressions for mn'' and an' by observing that as y varies from 0 to $mn + ret' - ret$, the value of mn'' varies from $mn + ret' - ret$ to 0 and the value of an' varies from an to $mn + an + ret' - ret$. This is captured by the inequalities $0 \leq mn'' \leq mn + ret' - ret$ and $an \leq an' \leq mn + an + ret' - ret$. However, only one of these is required to fully define the values of mn'' and an' because of the expression $mn'' + an' = mn + an + ret' - ret$. The variable y can be eliminated from Equation (6.23) in a similar way, resulting in the following expressions for V_d and V_e :

$$\begin{aligned}
 V_d &= \{M_{(3b,i),(0,ao,mn'',an',ret')}^{(MS,MR)} \mid mn'' + an' = mn + an + ret' - ret, 0 \leq mn'' \leq mn + ret' - ret, \\
 &\hspace{15em} ret \leq ret' \leq MR\} \quad (6.24)
 \end{aligned}$$

$$\begin{aligned}
 V_e &= \{M_{(2b,i),(0,ao,mn'',an',ret')}^{(MS,MR)} \mid mn'' + an' = mn + an + ret' - ret + 1, 0 \leq mn'' \leq mn + ret' - ret, \\
 &\hspace{15em} ret \leq ret' \leq MR\} \quad (6.25)
 \end{aligned}$$

Transition `mess_loss` (row 2 of Table 5.8) and `ack_loss` (rows 3 and 4 of Table 5.8) allow all markings in the downward-closed set of each marking in V_d to be reached. Similarly, rows 2, 4 and 5 of Table 5.6 allow the same for V_e . Applying Definition 30 to (6.24) and (6.25) to obtain the downward-closed sets of all markings in V_d and V_e gives:

$$\begin{aligned}
 V'_d &= DC(V_d) \\
 &= \{M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \mid M_{(3b,i),(0,ao,mn'',an',ret')}^{(MS,MR)} \in V_d, ao' \leq ao, mn''' \leq mn'', an'' \leq an'\} \\
 &= \{M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \mid ao' \leq ao, mn''' \leq mn'', an'' \leq an', mn'' + an' = mn + an \\
 &\hspace{10em} + ret' - ret, 0 \leq mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.26)
 \end{aligned}$$

$$\begin{aligned}
 V'_e &= DC(V_e) \\
 &= \{M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \mid M_{(2b,i),(0,ao,mn'',an',ret')}^{(MS,MR)} \in V_e, ao' \leq ao, mn''' \leq mn'', an'' \leq an'\} \\
 &= \{M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \mid ao' \leq ao, mn''' \leq mn'', an'' \leq an', mn'' + an' = mn + an \\
 &\hspace{10em} + ret' - ret + 1, 0 \leq mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.27)
 \end{aligned}$$

The variables mn'' and an' can be eliminated from Equations (6.26) and (6.27) by substituting the inequalities $mn''' \leq mn''$ and $an'' \leq an'$ wherever mn'' and an' appear, giving:

$$V'_d = \{M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \mid ao' \leq ao, mn''' + an'' \leq mn + an + ret' - ret, \\ 0 \leq mn''' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.28)$$

$$V'_e = \{M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \mid ao' \leq ao, mn''' + an'' \leq mn + an + ret' - ret + 1, \\ 0 \leq mn''' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.29)$$

Transition `receive_ack` (row 6 of Table 5.8) is enabled and can occur from any marking $M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_d$ in which $ao' = 0$ and $an'' > 0$, leading to a marking $M_{(4,i \oplus_{MS} 1),(mn''',an''-1,0,0,0)}^{(MS,MR)} \in V'_{(4,i \oplus_{MS} 1)}$. Also, `receive_ack` (row 7 of Table 5.6) is enabled by any marking $M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_e$ in which $ao' = 0$ and $an'' > 0$, leading to a marking $M_{(1,i \oplus_{MS} 1),(mn''',an''-1,0,0,0)}^{(MS,MR)} \in V'_{(1,i \oplus_{MS} 1)}$. The occurrence of the `receive_ack` transition from each enabling marking in V'_d and V'_e results in the sets of markings V_f (class 4) and V_g (class 1), respectively:

$$V_f = \{M_{(4,i \oplus_{MS} 1),(mn''',an''-1,0,0,0)}^{(MS,MR)} \mid M_{(3b,i),(0,0,mn''',an'',ret')}^{(MS,MR)} \in V'_d, an'' > 0\} \quad (6.30)$$

$$V_g = \{M_{(1,i \oplus_{MS} 1),(mn''',an''-1,0,0,0)}^{(MS,MR)} \mid M_{(2b,i),(0,0,mn''',an'',ret')}^{(MS,MR)} \in V'_e, an'' > 0\} \quad (6.31)$$

Using Equations (6.28) and (6.29) into (6.30) and (6.31) respectively gives:

$$V_f = \{M_{(4,i \oplus_{MS} 1),(mn''',an''',0,0,0)}^{(MS,MR)} \mid an''' = an'' - 1, 0 \leq mn''' + an'' \leq mn + an + ret' - ret, \\ 0 \leq mn''' \leq mn + ret' - ret, ret \leq ret' \leq MR, an'' > 0\} \quad (6.32)$$

$$V_g = \{M_{(1,i \oplus_{MS} 1),(mn''',an''',0,0,0)}^{(MS,MR)} \mid an''' = an'' - 1, 0 \leq mn''' + an'' \leq mn + an + ret' - ret + 1, \\ 0 \leq mn''' \leq mn + ret' - ret, ret \leq ret' \leq MR, an'' > 0\} \quad (6.33)$$

Substituting the expression $an'' = an''' + 1$ to eliminate an'' gives:

$$V_f = \{M_{(4,i \oplus_{MS} 1),(mn''',an''',0,0,0)}^{(MS,MR)} \mid 0 \leq mn''' + an''' \leq mn + an + ret' - ret - 1, \\ 0 \leq mn''' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.34)$$

$$V_g = \{M_{(1,i \oplus_{MS} 1),(mn''',an''',0,0,0)}^{(MS,MR)} \mid 0 \leq mn''' + an''' \leq mn + an + ret' - ret, \\ 0 \leq mn''' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.35)$$

An abstract view of the coverage of V'_d , V'_e , V_f and V_g over $V_i^{(MS,MR)}$ and $V_{(i \oplus_{MS} 1)}^{(MS,MR)}$ is given in Fig. 6.5. V'_d , V'_e , V_f and V_g are represented by the corresponding shaded portions of Fig. 6.5. V'_d is a subset of $V_{(3b,i)}^{(MS,MR)}$, V'_e is a subset of $V_{(2b,i)}^{(MS,MR)}$, V_f is a subset of $V_{(4,i \oplus_{MS} 1)}^{(MS,MR)}$, and V_g is a subset of $V_{(1,i \oplus_{MS} 1)}^{(MS,MR)}$. Markings in V'_d can reach markings in V'_e and vice versa by occurrences of `send_ack` and `receive_mess` respectively, while markings in V_f and V_g can be reached from markings in which no old acknowledgements exist in V'_d and V'_e respectively by occurrences of `receive_ack` (i.e. receiving a new acknowledgement). As in Fig. 6.4, the extent of coverage will depend upon the initial marking, $M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)}$, chosen.

At this point we conjecture that $V'_d \cup V'_e \cup V_f \cup V_g$ is the ϵ -closure of a marking $M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)}$. We must ensure that there are no additional markings reachable via ϵ moves:

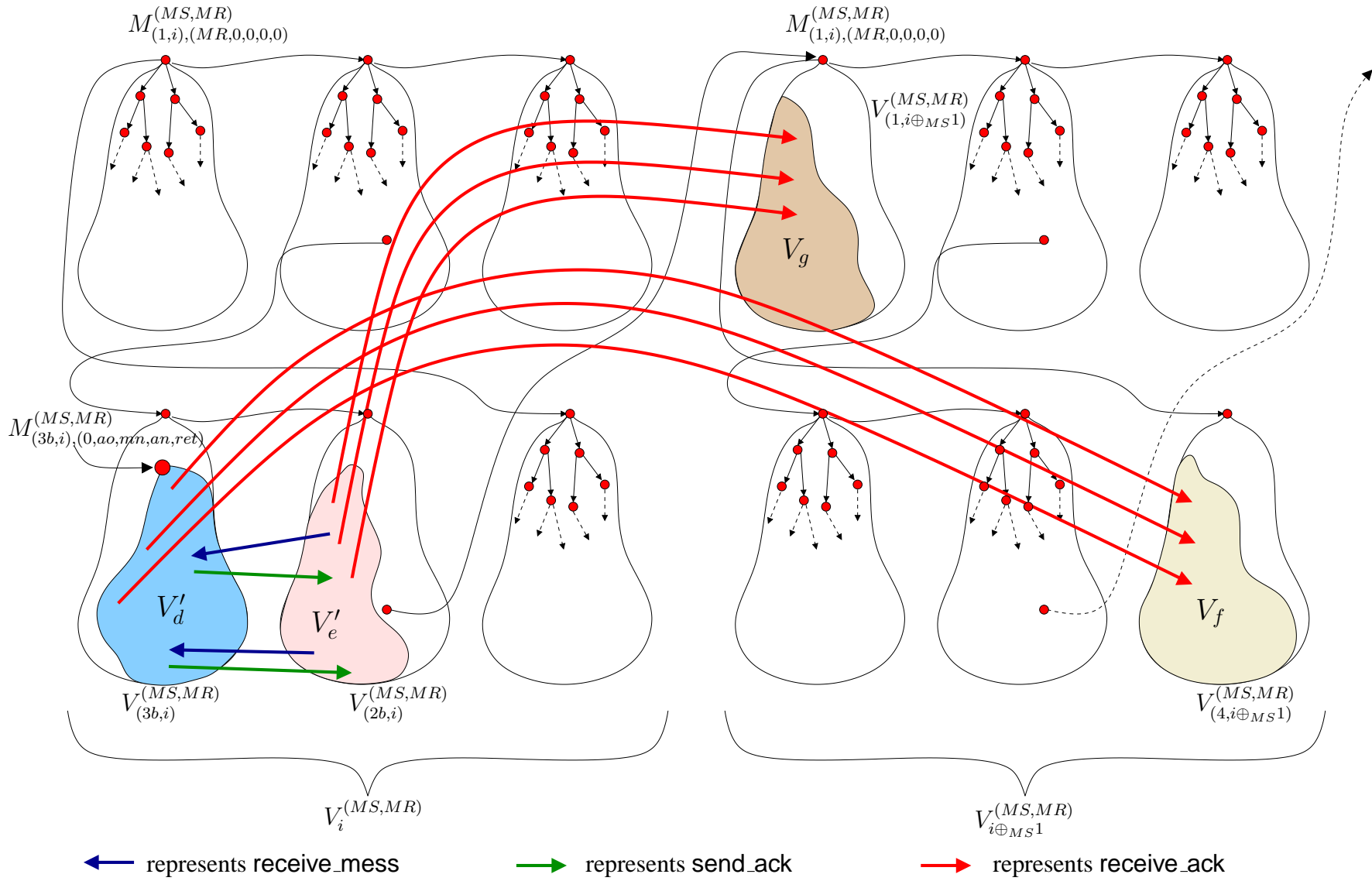


Figure 6.5: The coverage of V'_d , V'_e , V_f and V_g over $V_i^{(MS,MR)} \cup V_{i \oplus MS1}^{(MS,MR)}$.

Lemma 5. Any marking $M' \in V_{(MS,MR)}$ reachable via an ϵ move from a marking $M \in V'_d \cup V'_e \cup V_f \cup V_g$ is also contained in $V'_d \cup V'_e \cup V_f \cup V_g$, i.e. $\forall M \in V'_d \cup V'_e \cup V_f \cup V_g, M \xrightarrow{\epsilon} M' \implies M' \in V'_d \cup V'_e \cup V_f \cup V_g$.

Proof. For $M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_d$ at most 7 outgoing ϵ moves are defined by Table 5.8 and mapping *Prim*. They are treated systematically below.

mess_loss_new (row 2) is enabled by $M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_d$ if $mn''' > 0$. Its occurrence results in a marking $M_{(3b,i),(0,ao',mn'''-1,an'',ret')}^{(MS,MR)}$.

ack_loss_old and receive_dup_ack (rows 3 and 5) are enabled by $M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_d$ if $ao' > 0$. Its occurrence of either results in a marking $M_{(3b,i),(0,ao'-1,mn''',an'',ret')}^{(MS,MR)}$.

ack_loss_new (row 4) is enabled by $M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_d$ if $an'' > 0$. Its occurrence leads to a marking $M_{(3b,i),(0,ao',mn''',an''-1,ret')}^{(MS,MR)}$.

All three of these resulting markings are identical to their source marking, $M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_d$, except for either one less new message, one less old acknowledgement, or one less new acknowledgement. Because V'_d is a downward-closed set, and all three markings are the result of an action corresponding to loss, then by the definition of downward-closed sets (Definition 30), all three resulting markings are also elements of V'_d .

The remaining outgoing ϵ moves are a little more complex. treatment:

timeout_retrans (row 1) is enabled by $M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_d$ if $ret' < MR$. Its occurrence results in a marking $M_{(3b,i),(0,ao',mn'''+1,an'',ret'+1)}^{(MS,MR)}$. From the inequalities in Equation (6.28), $(mn''' + 1) + an'' \leq mn + an + (ret' + 1) - ret$, $0 < (mn''' + 1) \leq mn + (ret' + 1) - ret$, and $ret < (ret' + 1) \leq MR$ because of the restriction that $ret' < MR$. These satisfy the inequalities given in Equation (6.28) when substituting $(mn''' + 1)$ and $(ret' + 1)$ for mn''' and ret' respectively into Equation (6.28). Hence $M_{(3b,i),(0,ao',mn'''+1,an'',ret'+1)}^{(MS,MR)} \in V'_d$.

receive_ack (row 6) is enabled by $M_{(3b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_d$ if $ao' = 0$ and $an'' > 0$. Its occurrence leads to a marking $M_{(4,i \oplus_{MS} 1),(mn''',an''-1,0,0,0)}^{(MS,MR)}$. From the inequalities in Equation (6.28), $mn''' + (an'' - 1) \leq mn + an + ret' - ret - 1$. This matches the corresponding inequality in Equation (6.34) when substituting $(an'' - 1)$ for an''' in Equation (6.34). By inspection, the other inequalities from Equation (6.28) are the same as the corresponding inequalities in Equation (6.34). Hence $M_{(4,i \oplus_{MS} 1),(mn''',an''-1,0,0,0)}^{(MS,MR)} \in V_f$.

send_ack (row 7) is enabled by all markings in V'_d . Its occurrence leads to a marking $M_{(2b,i),(0,ao',mn''',an''+1,ret')}^{(MS,MR)}$. From the inequalities in Equation (6.28), $mn''' + (an'' + 1) \leq mn + an + ret' - ret + 1$. This matches the corresponding inequality in Equation (6.29) when substituting $(an'' + 1)$ for an''' in Equation (6.29). By inspection, the other inequalities from Equation (6.28) are the same as the corresponding inequalities in (6.29). Hence $M_{(2b,i),(0,ao',mn''',an''+1,ret')}^{(MS,MR)} \in V'_e$.

For $M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_e$ at most 7 outgoing ϵ moves are defined by Table 5.6 and mapping *Prim*. As before, we consider them one by one.

mess_loss_new (row 2) is enabled by $M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V'_e$ if $mn''' > 0$. Its occurrence results in a marking $M_{(2b,i),(0,ao',mn'''-1,an'',ret')}^{(MS,MR)}$.

ack_loss_old and receive_dup_ack (rows 4 and 6) are enabled by $M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V_e'$ if $ao' > 0$. Its occurrence of either results in the marking $M_{(2b,i),(0,ao'-1,mn''',an'',ret')}^{(MS,MR)}$.

ack_loss_new (row 5) is enabled by $M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V_e'$ if $an'' > 0$. Its occurrence results in a marking $M_{(2b,i),(0,ao',mn''',an''-1,ret')}^{(MS,MR)}$.

All three of these resulting markings are identical to their source marking, $M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V_e'$, except for either one less new message, one less old acknowledgement, or one less new acknowledgement. Because V_e' is a downward-closed set, then by the definition of downward-closed sets (Definition 30), all three resulting markings are also elements of V_e' .

The remaining three outgoing ϵ moves require more careful treatment:

timeout_retrans (row 1) is enabled by $M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V_e'$ if $ret' < MR$. Its occurrence results in a marking $M_{(2b,i),(0,ao',mn'''+1,an'',ret'+1)}^{(MS,MR)}$. From the inequalities in Equation (6.29), $(mn'''+1) + an'' \leq mn+an+(ret'+1)-ret+1$, $0 < (mn'''+1) \leq mn+(ret'+1)-ret$, and $ret < (ret'+1) \leq MR$ because of the restriction that $ret' < MR$. These satisfy the inequalities given in Equation (6.29) when substituting $(mn'''+1)$ and $(ret'+1)$ for mn''' and ret' respectively into Equation (6.29). Hence $M_{(2b,i),(0,ao',mn'''+1,an'',ret'+1)}^{(MS,MR)} \in V_e'$.

receive_mess (row 3) is enabled by $M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V_e'$ if $mn''' > 0$. Its occurrence results in a marking $M_{(3b,i),(0,ao',mn'''-1,an'',ret')}^{(MS,MR)}$. From the inequalities in Equation (6.29), $(mn'''-1) + an'' \leq mn+an+ret'-ret$, and $0 \leq (mn'''-1) < mn+ret'-ret$ because of the restriction that $mn''' > 0$. These satisfy the corresponding inequalities in Equation (6.28) when substituting $(mn'''-1)$ for mn''' in Equation (6.28). By inspection, the other two inequalities from Equation (6.29) are the same as the corresponding inequalities in Equation (6.28). Hence $M_{(3b,i),(0,ao',mn'''-1,an'',ret')}^{(MS,MR)} \in V_d'$.

receive_ack (row 7) is enabled by $M_{(2b,i),(0,ao',mn''',an'',ret')}^{(MS,MR)} \in V_e'$ if $ao' = 0$ and $an'' > 0$. Its occurrence results in a marking $M_{(1,i \oplus_{MS} 1),(mn''',an''-1,0,0,0)}^{(MS,MR)}$. From the inequalities in Equation (6.29), $mn''' + (an'' - 1) \leq mn+an+ret'-ret$. This matches the corresponding inequality in Equation (6.35) when substituting $(an'' - 1)$ for an''' into Equation (6.35). By inspection, the other two relevant inequalities from Equation (6.29) match the corresponding inequalities in Equation (6.35). Hence $M_{(1,i \oplus_{MS} 1),(mn''',an''-1,0,0,0)}^{(MS,MR)} \in V_g$.

For $M_{(4,i \oplus_{MS} 1),(mn''',an''',0,0,0)}^{(MS,MR)} \in V_f$ at most 4 outgoing ϵ moves are defined by Table 5.9 and mapping *Prim*, substituting $i \oplus_{MS} 1$ for i . Three of them correspond to loss events:

mess_loss_old (row 2) is enabled by $M_{(4,i \oplus_{MS} 1),(mn''',an''',0,0,0)}^{(MS,MR)} \in V_f$ if $mn''' > 0$. Its occurrence results in a marking $M_{(4,i \oplus_{MS} 1),(mn'''-1,an''',0,0,0)}^{(MS,MR)}$.

ack_loss_old and receive_dup_ack (rows 3 and 4) are enabled by $M_{(4,i \oplus_{MS} 1),(mn''',an''',0,0,0)}^{(MS,MR)} \in V_f$ if $an''' > 0$. Its occurrence of either leads to a marking $M_{(4,i \oplus_{MS} 1),(mn''',an'''-1,0,0,0)}^{(MS,MR)}$.

Both of these resulting markings are identical to $M_{(4,i \oplus_{MS} 1),(mn''',an''',0,0,0)}^{(MS,MR)} \in V_f$ except for either one less old message or one less old acknowledgement. From Equation (6.34), we can see that V_f is already a downward-closed set, without needing to apply the *DC* operator. Hence, because V_f is a downward-closed set, from the definition of downward closed sets (Definition 30), both of these resulting markings are also elements of V_f .

The fourth outgoing ϵ move corresponds to **send_ack**:

send_ack (row 5) is enabled by all markings in V_f . Its occurrence results in a marking $M_{(1,i \oplus_{MS} 1), (mn''', an'''+1, 0, 0, 0)}^{(MS, MR)}$. From the inequalities in Equation (6.34), $0 \leq mn''' + (an''' + 1) \leq mn + an + ret' - ret$. This matches the corresponding inequality in Equation (6.35) when substituting $(an''' + 1)$ for an''' in Equation (6.35). By inspection, the other two inequalities from Equation (6.34) are the same as the corresponding inequalities in Equation (6.35). Hence $M_{(1,i \oplus_{MS} 1), (mn''', an'''+1, 0, 0, 0)}^{(MS, MR)} \in V_g$.

In a nearly identical fashion to $M_{(4,i \oplus_{MS} 1), (mn''', an''', 0, 0, 0)}^{(MS, MR)} \in V_f$, $M_{(1,i \oplus_{MS} 1), (mn''', an''', 0, 0, 0)}^{(MS, MR)} \in V_g$ also has at most 4 outgoing ϵ moves, defined by Table 5.4 and mapping *Prim*, substituting $i \oplus_{MS} 1$ for i . Three of the outgoing ϵ moves also correspond to loss, with destination markings remaining in V_g , and the arguments put forward as to why this is the case are identical to the corresponding three outgoing ϵ moves from V_f , hence are not repeated. The only other outgoing ϵ move corresponds to **receive_mess** on row 3 of Table 5.4. This transition is enabled by $M_{(1,i \oplus_{MS} 1), (mn''', an''', 0, 0, 0)}^{(MS, MR)} \in V_g$ if $mn''' > 0$. Its occurrence results in a marking $M_{(4,i \oplus_{MS} 1), (mn'''-1, an''', 0, 0, 0)}^{(MS, MR)}$. From the inequalities in Equation (6.35), $0 \leq (mn''' - 1) + an''' \leq mn + an + ret' - ret - 1$ and $0 \leq (mn''' - 1) < mn + ret' - ret$ because of the restriction that $mn''' > 0$. These satisfy the corresponding inequalities in Equation (6.34) when substituting $(mn''' - 1)$ for mn''' in Equation (6.34). By inspection, the remaining inequality (on *ret*) from Equation (6.35) matches the corresponding inequality in Equation (6.34). Hence $M_{(4,i \oplus_{MS} 1), (mn'''-1, an''', 0, 0, 0)}^{(MS, MR)} \in V_f$.

All successors of all markings in $V'_d \cup V'_e \cup V_f \cup V_g$ reachable by an ϵ move have now been explored, and all are contained in $V'_d \cup V'_e \cup V_f \cup V_g$. Thus the lemma is proved. \square

Corollary 2. *The ϵ -closure of $M_{(3b,i), (0, ao, mn, an, ret)}^{(MS, MR)}$, $Closure(M_{(3b,i), (0, ao, mn, an, ret)}^{(MS, MR)})$, is given by the union of the parameterised sets $V'_d \cup V'_e \cup V_f \cup V_g$ defined by Equations (6.28), (6.29), (6.34) and (6.35) respectively, i.e. (replacing double and triple primes with single primes):*

$$\begin{aligned}
 Closure(M_{(3b,i), (0, ao, mn, an, ret)}^{(MS, MR)}) &= \{M_{(3b,i), (0, ao', mn', an', ret')}^{(MS, MR)} \mid ao' \leq ao, \\
 &\quad mn' + an' \leq mn + an + ret' - ret, 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \\
 \cup \{M_{(2b,i), (0, ao', mn', an', ret')}^{(MS, MR)} \mid &ao' \leq ao, mn' + an'' \leq mn + an + ret' - ret + 1, \\
 &\quad 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \\
 \cup \{M_{(4,i \oplus_{MS} 1), (mn', an', 0, 0, 0)}^{(MS, MR)} \mid &mn' + an' \leq mn + an + ret' - ret - 1, \\
 &\quad 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \\
 \cup \{M_{(1, \oplus_{MS} 1), (mn', an', 0, 0, 0)}^{(MS, MR)} \mid &mn' + an' \leq mn + an + ret' - ret, \\
 &\quad 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR\} \quad (6.36)
 \end{aligned}$$

The only transitions enabled by the markings in $Closure(M_{(3b,i), (0, ao, mn, an, ret)}^{(MS, MR)})$ that do not map to ϵ are given by Row 1 of Table 5.4 and Row 1 of Table 5.9. They both map to the **Send** service primitive. From Row 1 of Table 5.4 and Row 1 of Table 5.9 the corresponding set of non- ϵ -labelled outgoing edges from all markings in the above closure is given by:

$$\begin{aligned}
 OutEdges(Closure(M_{(3b,i), (0, ao, mn, an, ret)}^{(MS, MR)})) &= \{(M_{(4,i \oplus_{MS} 1), (mn', an', 0, 0, 0)}^{(MS, MR)}, \mathbf{Send}, \\
 &\quad M_{(3a,i \oplus_{MS} 1), (mn', an', 1, 0, 0)}^{(MS, MR)}) \mid M_{(4,i \oplus_{MS} 1), (mn', an', 0, 0, 0)}^{(MS, MR)} \in V_f\} \\
 \cup \{(M_{(1,i \oplus_{MS} 1), (mn', an', 0, 0, 0)}^{(MS, MR)}, \mathbf{Send}, &M_{(2a,i \oplus_{MS} 1), (mn', an', 1, 0, 0)}^{(MS, MR)}) \\
 &\quad \mid M_{(1,i \oplus_{MS} 1), (mn', an', 0, 0, 0)}^{(MS, MR)} \in V_g\} \quad (6.37)
 \end{aligned}$$

Using the restrictions required for V_f and V_g from Equations (6.34) and (6.35) in Equation (6.37) gives:

$$\begin{aligned}
& OutEdges(Closure(M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)})) = \\
& \{ (M_{(4,i\oplus_{MS1}),(mn',an',0,0,0)}^{(MS,MR)}, \text{Send}, M_{(3a,i\oplus_{MS1}),(mn',an',1,0,0)}^{(MS,MR)} \mid mn' + an' \leq mn + an \\
& \quad + ret' - ret - 1, 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR \} \\
& \cup \{ (M_{(1,i\oplus_{MS1}),(mn',an',0,0,0)}^{(MS,MR)}, \text{Send}, M_{(2a,i\oplus_{MS1}),(mn',an',1,0,0)}^{(MS,MR)} \mid mn' + an' \leq mn + an \\
& \quad + ret' - ret, 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR \} \quad (6.38)
\end{aligned}$$

6.4 Determinisation

Now that we have expressions for the ϵ -closures that we require, determinisation of the FSA interpretation of $RG_{(MS,MR)}$, $FSA_{RG_{(MS,MR)}}$, to obtain an equivalent deterministic FSA, $DFSA_{RG_{(MS,MR)}}$, using lazy subset construction, proceeds as described in [41] using the formalism from Section 2.4.

Recall from Section 2.4 that a deterministic FSA, $DFSA$, can be defined as $DFSA = (S^{det}, \Sigma, \Delta^{det}, s_0^{det}, F^{det})$. The initial state of $DFSA_{RG_{(MS,MR)}}$ is the ϵ -closure of the initial marking of $FSA_{RG_{(MS,MR)}}$, i.e. $Closure(M_{(1,0),(0,0,0,0,0)}^{(MS,MR)})$. $Closure(M_{(1,0),(0,0,0,0,0)}^{(MS,MR)})$, which we denote C_0 , equals $\{M_{(1,0),(0,0,0,0,0)}^{(MS,MR)}\}$ as there are no outgoing ϵ edges from the initial state of $FSA_{RG_{(MS,MR)}}$. This is readily evident from the initial marking of $CPN_{(MS,MR)}$, as only the `send_mess` transition (corresponding to the `Send` service primitive) is enabled in the initial marking, regardless of the value of the parameters MS and MR . Thus

$$C_0 = s_0^{det} = Closure(s_0) = Closure(M_{(1,0),(0,0,0,0,0)}^{(MS,MR)}) = \{M_{(1,0),(0,0,0,0,0)}^{(MS,MR)}\} \quad (6.39)$$

Furthermore, $C_0 \in S^{det}$.

Proceeding with lazy subset evaluation, the single state in C_0 has only one outgoing non- ϵ arc defined. This corresponds to row 1 of Table 5.4 which maps to the `Send` service primitive. It leads to the state $M_{(2a,0),(0,0,1,0,0)}^{(MS,MR)}$. Note that no old duplicates can exist in the underlying channels at this point, because this is the first message being sent and thus no messages have been sent previously.

The ϵ -closure of $M_{(2a,0),(0,0,1,0,0)}^{(MS,MR)}$ is given by evaluating Equation (6.15) for $M_{(2a,0),(0,0,1,0,0)}^{(MS,MR)}$, which we denote C_1 :

$$\begin{aligned}
C_1 = Closure(M_{(2a,0),(0,0,1,0,0)}^{(MS,MR)}) = \{ & M_{(2a,0),(mo'',ao'',mn'',0,ret')}^{(MS,MR)} \mid 0 \leq mo'' + ao'' \leq 0, 0 \leq mo'' \leq 0, \\
& 0 \leq mn'' \leq 1 + ret', 0 \leq ret' \leq MR \} \quad (6.40)
\end{aligned}$$

There are no class 3a markings because there are no old messages ($mo = 0$) and hence $mo \not\geq 0$. Simplifying Equation (6.40) and dropping the primes gives:

$$C_1 = \{M_{(2a,0),(0,0,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mn \leq 1 + ret, 0 \leq ret \leq MR\} \quad (6.41)$$

This result makes sense intuitively, as from the initial state we can get to any state in which the original message plus up to MR retransmissions have been sent, and then may have been lost. Thus $C_1 \in S^{det}$ and $(s_0^{det}, \text{Send}, C_1) \in \Delta^{det}$. s_0^{det} and C_1 are illustrated in Fig. 6.6. s_0^{det} is shown as a large red circle within $V_{(1,0)}^{(MS,MR)}$, which leads to the successor $C_1 \in V_{(2a,0)}^{(MS,MR)}$ via the `Send` primitive. The set C_1 covers only some of the markings in $V_{(2a,0)}^{(MS,MR)}$ and so has been depicted to reflect this.

From Table 5.5 and Equation (6.17) the only outgoing non-epsilon edges of states in C_1 are edges labelled by `Receive`, corresponding to row 5 of this table. From the expression on this row the successor of a state

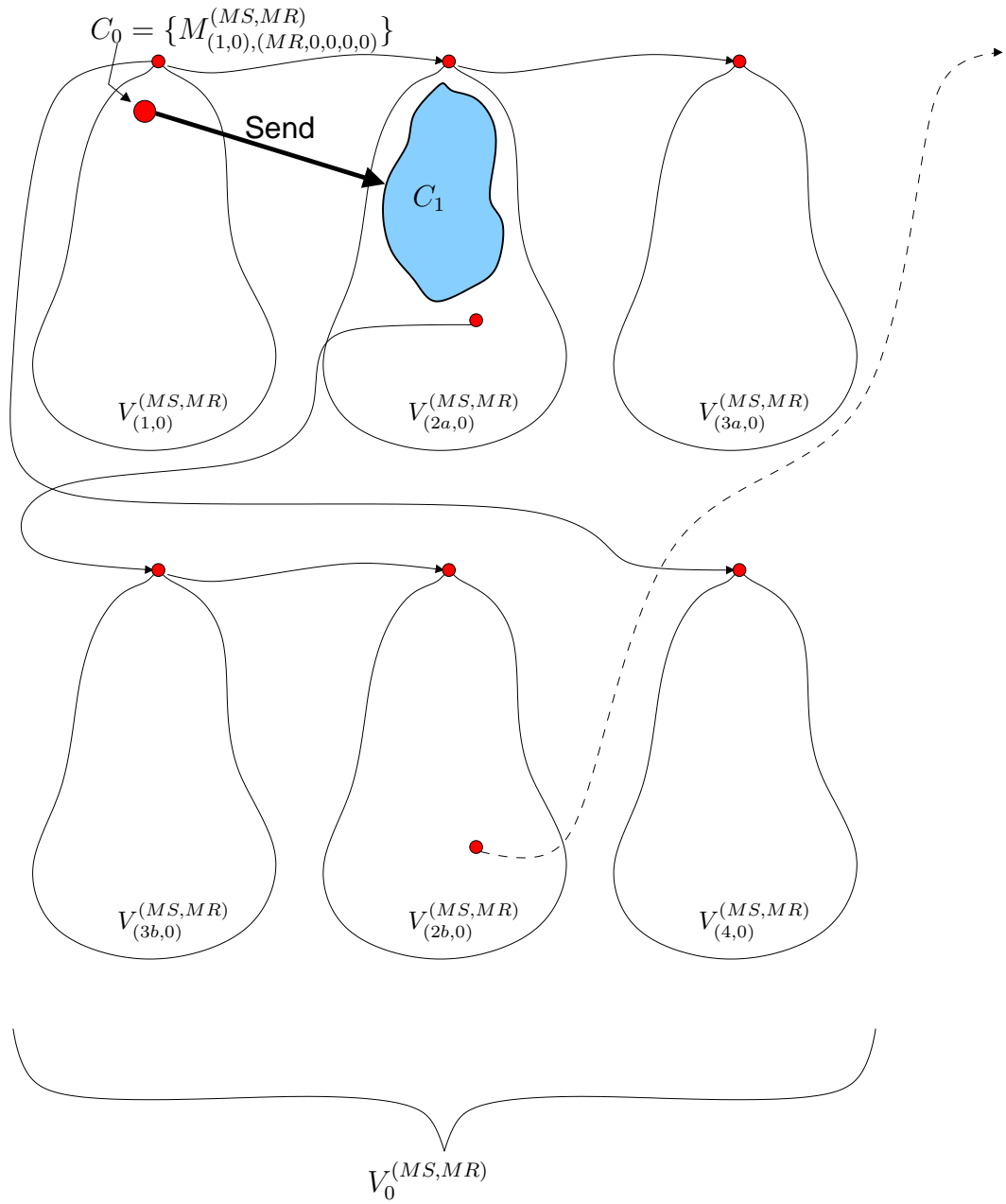


Figure 6.6: Construction of a deterministic FSA using subset construction with lazy evaluation, showing the initial state, s_0^{det} , and its only successor, C_1 .

$M_{(2a,0),(0,0,mn,0,ret)}^{(MS,MR)}$ is $M_{(3b,0),(0,0,mn-1,0,ret)}^{(MS,MR)}$ but this successor only exists if $mn > 0$, and thus the direct successors of all states in C_1 are captured in V_h :

$$\begin{aligned} V_h &= \{M_{(3b,0),(0,0,mn-1,0,ret)}^{(MS,MR)} \mid M_{(2a,0),(0,0,mn,0,ret)}^{(MS,MR)} \in C_1, mn > 0\} \\ &= \{M_{(3b,0),(0,0,mn-1,0,ret)}^{(MS,MR)} \mid 0 < mn \leq ret + 1, 0 \leq ret \leq MR\} \end{aligned}$$

According to the procedure in Section 2.4 the successor of C_1 is the union of the ϵ -closures of all markings in V_h , i.e. $CLOSURE(V_h)$. However, rather than calculate the ϵ -closure of all states in V_h , let us investigate the ϵ -closure of the state $M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)} \in V_h$, i.e. when $mn - 1 = ret = 0$.

The ϵ -closure of $M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}$ is given by evaluating Equation (6.36) for $M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}$, which we denote C_2 :

$$\begin{aligned}
 C_2 = \text{Closure}(M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}) &= \{M_{(3b,0),(0,ao,mn,an,ret)}^{(MS,MR)} \mid ao \leq 0, 0 \leq mn + an \leq ret, \\
 &\quad 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \\
 \cup \{M_{(2b,0),(0,ao,mn,an,ret)}^{(MS,MR)} \mid &ao \leq 0, 0 \leq mn + an \leq ret + 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \\
 \cup \{M_{(4,1),(mn,an,0,0,0)}^{(MS,MR)} \mid &0 \leq mn + an \leq ret - 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \\
 \cup \{M_{(1,1),(mn,an,0,0,0)}^{(MS,MR)} \mid &0 \leq mn + an \leq ret, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \quad (6.42)
 \end{aligned}$$

Now that concrete values for mo, ao, mn, an and ret have been given, the inequality $0 \leq mn''' \leq ret'$ from the third set in the union in Equation (6.42) becomes redundant. This is because the inequality $mn''' + an''' \leq ret' - 1$ limits the range of values of mn''' to $0 \leq mn''' \leq ret' - 1$ (recall that $mn''' \geq 0, an''' \geq 0$ is implicit). The inequality $0 \leq mn''' \leq ret'$ from the first and fourth sets in the union in Equation (6.42) is also redundant. Equation (6.42) can be further simplified by the realisation that for the third set in the union, $mn''' + an''' \leq MR - 1$ covers all possible values of mn''' and an''' given by the inequalities $mn''' + an''' \leq ret' - 1, 0 \leq ret' \leq MR$ (again, $mn''' \geq 0$ and $an''' \geq 0$ is implicit). The same is true of $mn''' + an''' \leq MR$ for the inequalities $mn''' + an''' \leq ret', 0 \leq ret' \leq MR$ in the fourth set in the union. Given that ret' plays no other part in the definition of the third and fourth sets in the union, ret' can be eliminated from these two sets. Applying these simplifications to Equation (6.42) gives:

$$\begin{aligned}
 C_2 = \{M_{(3b,0),(0,0,mn,an,ret)}^{(MS,MR)} \mid &0 \leq mn + an \leq ret, 0 \leq ret \leq MR\} \\
 \cup \{M_{(2b,0),(0,0,mn,an,ret)}^{(MS,MR)} \mid &0 \leq mn + an \leq ret + 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \\
 \cup \{M_{(4,1),(mn,an,0,0,0)}^{(MS,MR)} \mid &0 \leq mn + an \leq MR - 1\} \\
 \cup \{M_{(1,1),(mn,an,0,0,0)}^{(MS,MR)} \mid &0 \leq mn + an \leq MR\} \quad (6.43)
 \end{aligned}$$

We show that $\text{Closure}(M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}) = \text{CLOSURE}(V_h)$ in the following lemma.

Lemma 6. $C_2 = \text{CLOSURE}(V_h)$.

Proof. The class 3b markings in C_2 span all allowable combinations of the variables mn, an and ret from the definition of class 3b markings in Table 5.3, for the case $ao = 0$. Given that there are no old acknowledgements in any of the states in V_h , and that from Equation (6.36) the number of old acknowledgements of the argument marking only affects the number of old acknowledgements in the markings of its ϵ -closure (ao' appears only in the inequality $ao' \leq ao$) then C_2 must include all possible class 3b markings from the ϵ -closure of any marking in V_h .

The argument is the same for the class 2b markings in C_2 . All allowable combinations of the variables mn, an and ret from the definition of class 2b markings in Table 5.3 are present in C_2 . An identical argument on the number of old acknowledgements holds for the class 2b markings as did for the class 3b markings.

The argument for the class 4 and 1 markings in C_2 is even simpler. C_2 contains class 1 and class 4 markings that cover *all possible* class 1 and 4 markings as defined in Table 5.3 for $i = 1$. It is therefore not possible to have excluded any class 1 or class 4 marking from C_2 , and hence C_2 must include all class 1 and class 4 markings in the ϵ -closure of any marking in V_h .

Hence, because $M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)} \in V_h$, we conclude that $\text{CLOSURE}(V_h) = C_2$ and thus the lemma is proved. \square

Corollary 3. $C_2 \in S^{det}$ and $(C_1, \text{Receive}, C_2) \in \Delta^{det}$

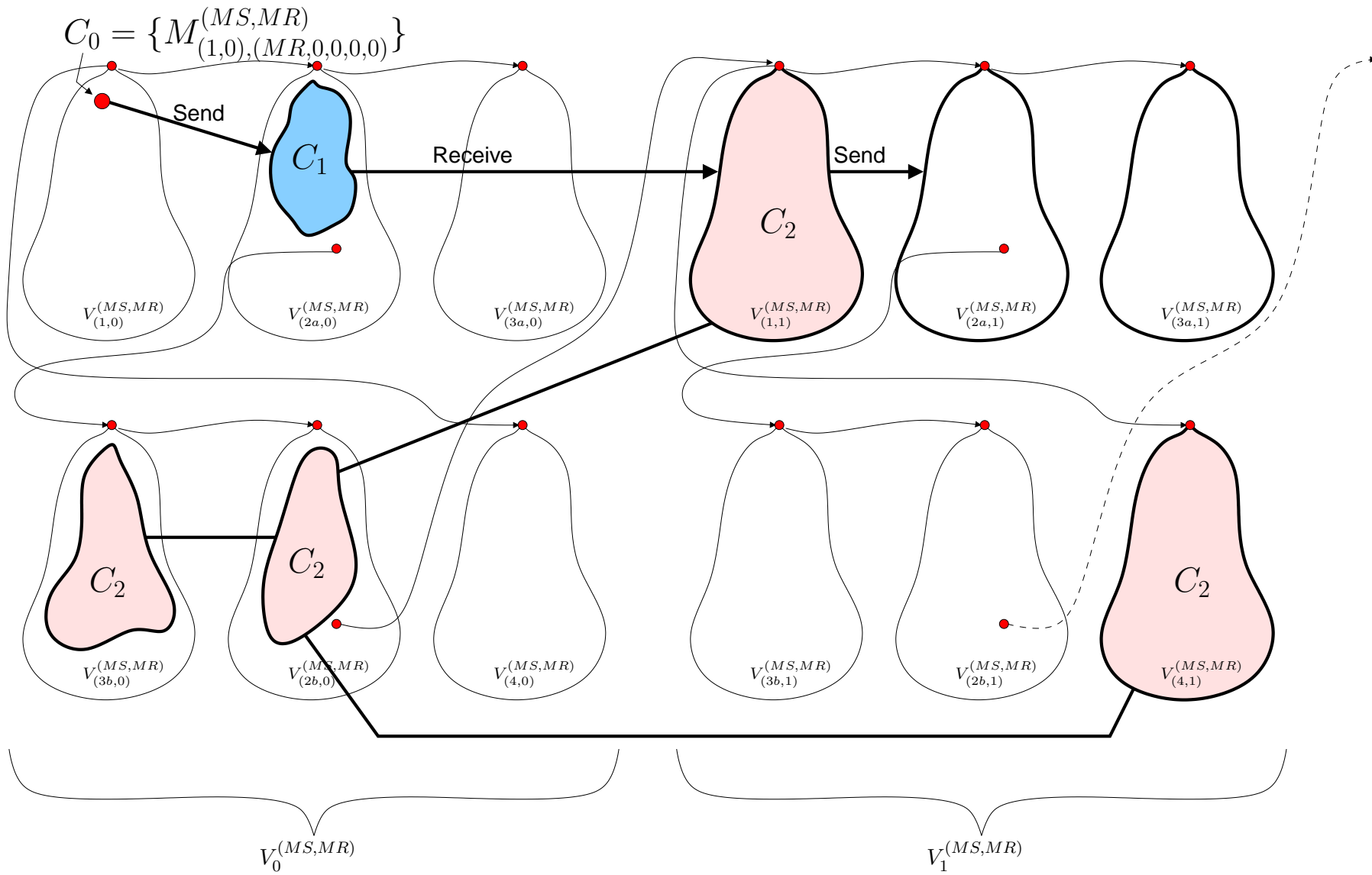


Figure 6.7: Construction of a deterministic FSA showing the addition of C_2 , the successor of C_1 from Fig. 6.6.

This result is illustrated in Fig. 6.7. Note that although the set C_2 comprises states from $V_{(2b,0)}^{(MS,MR)}$, $V_{(3b,0)}^{(MS,MR)}$, $V_{(1,1)}^{(MS,MR)}$ and $V_{(4,1)}^{(MS,MR)}$, the states in all four of these subsets in Fig. 6.7 are part of the single successor state of C_1 . Hence, in Fig. 6.7, the four shaded subsets representing C_2 are connected via solid lines with no arrowheads. Note that to reflect Equation (6.43), the depiction of the subsets of C_2 in $V_{(3b,0)}^{(MS,MR)}$ and $V_{(2b,0)}^{(MS,MR)}$ do not cover all of the markings in $V_{(3b,0)}^{(MS,MR)}$ or $V_{(2b,0)}^{(MS,MR)}$, whereas the subsets of C_2 in $V_{(1,1)}^{(MS,MR)}$ and $V_{(4,1)}^{(MS,MR)}$ do.

The only outgoing non- ϵ arcs from states in C_2 are from the class 1 and class 4 markings, corresponding to the **Send** primitive, and are given by Row 1 of both Table 5.4 and Table 5.9. In the CPN these correspond to the **send_mess** transition, able to occur from all class 1 and class 4 markings. All outgoing arcs from class 2b and 3b markings in $RG_{(MS,MR)}$ map to ϵ moves (and hence their destination states are also in C_2).

Because $CLOSURE(V_h) = C_2$ then $OutEdges(C_2) = OutEdges(CLOSURE(V_h)) = OutEdges(Closure(M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}))$. Hence, evaluating Equation (6.38) for $M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}$ gives all outgoing non- ϵ arcs from the markings in C_2 .

As an aside, we need not have chosen the marking $M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}$ for this purpose. We may have chosen any $M_{(3b,0),(0,0,mn,0,ret)}^{(MS,MR)} \in V_h$ provided $mn = ret$. The reason for this can be seen in Equation (6.36). The number of new messages in a class 3b or 2b marking become the number of old messages in a class 1 or 4 marking, because of the incremented sender sequence number. Provided the closure passed to $OutEdges$ contains a class 3b and class 2b marking with the maximum allowable number of old messages, then all allowable class 1 and class 4 markings (with sender sequence number equal to 1) will be covered. The condition $mn = ret$ implies that no new messages have yet been lost, and we can still retransmit another $MR - ret$ times, thus a class 3b and 2b marking with the maximum allowable value of $mn = MR$ can be reached. Because it is only the class 1 and class 4 markings from C_2 that have outgoing non- ϵ arcs, then this guarantees that all outgoing non- ϵ arcs from markings in C_2 are obtained.

For consistency, however, we retain the choice of $Closure(M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)})$ as the argument to $OutEdges$. The resulting arcs, given by Equation (6.38), have successor states given by V_j :

$$\begin{aligned} V_j &= \{s' \mid (s, \mathbf{Send}, s') \in OutEdges(Closure(M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}))\} \\ &= \{M_{(3a,0 \oplus MS1),(mn,an,1,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret - 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \\ &\cup \{M_{(2a,0 \oplus MS1),(mn,an,1,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \end{aligned} \quad (6.44)$$

Performing similar simplifications to those performed in Equation (6.43), Equation (6.44) can be simplified to become:

$$\begin{aligned} V_j &= \{M_{(3a,1),(mn,an,1,0,0)}^{(MS,MR)} \mid mn + an \leq MR - 1, MR > 0\} \\ &\cup \{M_{(2a,1),(mn,an,1,0,0)}^{(MS,MR)} \mid mn + an \leq MR\} \end{aligned} \quad (6.45)$$

According to the procedure in Section 2.4 the successor of C_2 is the union of the ϵ -closures of all markings in V_j , i.e. $CLOSURE(V_j)$. Let us leave the concrete domain at this point. We are now in a situation where we can have both old messages and old acknowledgements in the channels, and we can see from Fig. 6.7 that C_2 spans *all* markings in $V_{(1,1)}^{(MS,MR)}$ and $V_{(4,1)}^{(MS,MR)}$. Consider the set of states given by $V_{(k,i)}$, when replacing the sender sequence number of 1 by i in Equation (6.45):

$$\begin{aligned} V_{(k,i)} &= \{M_{(3a,i),(mo,ao,1,0,0)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1\} \\ &\cup \{M_{(2a,i),(mo,ao,1,0,0)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR\} \end{aligned} \quad (6.46)$$

for each $i, 0 \leq i \leq MR$. This gives a family of sets of markings, each identical to V_j apart from the sender sequence number. When $i = 1$ we have $V_{(k,1)} = V_j$.

In order to discover the ϵ -closure of $V_{(k,i)}$, $CLOSURE(V_{(k,i)})$, we use a similar procedure to that used to discover the ϵ -closure, C_2 . We select an appropriate single marking in $V_{(k,i)}$, calculate its ϵ -closure, and demonstrate that it covers all possible markings that could be in the ϵ -closure of any marking from $V_{(k,i)}$. The marking we have selected is the marking $M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)}$, because it has the most number of old messages, least number of new messages, and least number of retransmissions out of all markings in $V_{(k,i)}$, and so intuition suggests that its ϵ -closure will span more markings than any other marking in $V_{(k,i)}$.

The ϵ -closure of $M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)}$ is given by evaluating Equation (6.15) for $M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)}$, resulting in a set of markings (specified in terms of i) which we denote $C_{(3,i)}$:

$$\begin{aligned} C_{(3,i)} = & \{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mo \leq MR, 0 \leq mn \leq 1 + ret, \\ & 0 \leq ret \leq MR\} \\ & \cup \{M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mo \leq MR - 1, 0 \leq mn \leq 1 + ret, \\ & 0 \leq ret \leq MR, MR > 0\} \end{aligned} \quad (6.47)$$

where the appearance of i in the subscript shows that $C_{(3,i)}$ is parametric. Simplifying Equation (6.47) to remove redundant inequalities gives:

$$\begin{aligned} C_{(3,i)} = & \{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq 1 + ret, 0 \leq ret \leq MR\} \\ & \cup \{M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq 1 + ret, 0 \leq ret \leq MR\} \end{aligned} \quad (6.48)$$

where we drop the redundant $MR > 0$ term. By inspection, this ϵ -closure covers every class 2a and 3a marking defined in rows 2 and 4 of Table 5.3, for $i \in \{0, 1, \dots, MS\}$, i.e.

$$C_{(3,i)} = V_{(2a,i)}^{(MS,MR)} \cup V_{(3a,i)}^{(MS,MR)} \quad (6.49)$$

Lemma 7. $C_{(3,i)} = CLOSURE(V_{(k,i)})$.

Proof. From Tables 5.5 and 5.7 and the mapping *Prim*, the destination marking of every outgoing ϵ -labelled arc with its source marking in $C_{(3,i)}$ is also in $C_{(3,i)}$, i.e. $CLOSURE(C_{(3,i)}) = C_{(3,i)}$. The only outgoing arcs of states in $C_{(3,i)}$ that have destinations that are not in $C_{(3,i)}$ are labelled by the **Receive** service primitive, specifically, the arcs defined in row 5 of Table 5.5. They correspond to the occurrence of the `receive_mess` transition, moving from a class 2a to a class 3b marking.

Because $V_{(k,i)}$ contains only class 2a and 3a markings with a sender sequence number of i , then closure of all markings in $V_{(k,i)}$, $CLOSURE(V_{(k,i)})$, will contain only class 2a and 3a markings with a sender sequence number of i . Because $C_{(3,i)}$ is the ϵ -closure of $M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)} \in V_{(k,i)}$ and $C_{(3,i)}$ contains all class 2a and 3a markings with a sender sequence number of i , the lemma is proved. \square

Corollary 4. From Lemma 7 when $i = 1$, $C_{(3,1)} \in S^{det}$ and hence $(C_2, \text{Send}, C_{(3,1)}) \in \Delta^{det}$.

We illustrate this result in Fig. 6.8, which shows $C_{(3,1)}$ covering all markings in $V_{(2a,1)}^{(MS,MR)}$ and $V_{(3a,1)}^{(MS,MR)}$, and the arc from C_2 to $C_{(3,1)}$ labelled with the **Send** service primitive.

As stated in the proof of Lemma 7, the only outgoing non- ϵ edges from states in $C_{(3,i)}$ are labelled by **Receive**, corresponding to the `receive_mess` transition occurring from a class 2a marking (row 5 in Table 5.5). Because $C_{(3,i)} = Closure(M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)})$ this means that $OutEdges(Closure(M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)}))$ will give the successors of all outgoing non- ϵ edges from states in $C_{(3,i)}$. By making the appropriate substitutions into Equation 6.17, these are given by $V_{(l,i)}$:

$$\begin{aligned} V_{(l,i)} = & \{s' \mid (s, \text{Receive}, s') \in OutEdges(Closure(M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)}))\} \\ = & \{M_{(3b,i),(0,ao,mn-1,0,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 < mn \leq 1 + ret, 0 \leq ret \leq MR\} \end{aligned} \quad (6.50)$$

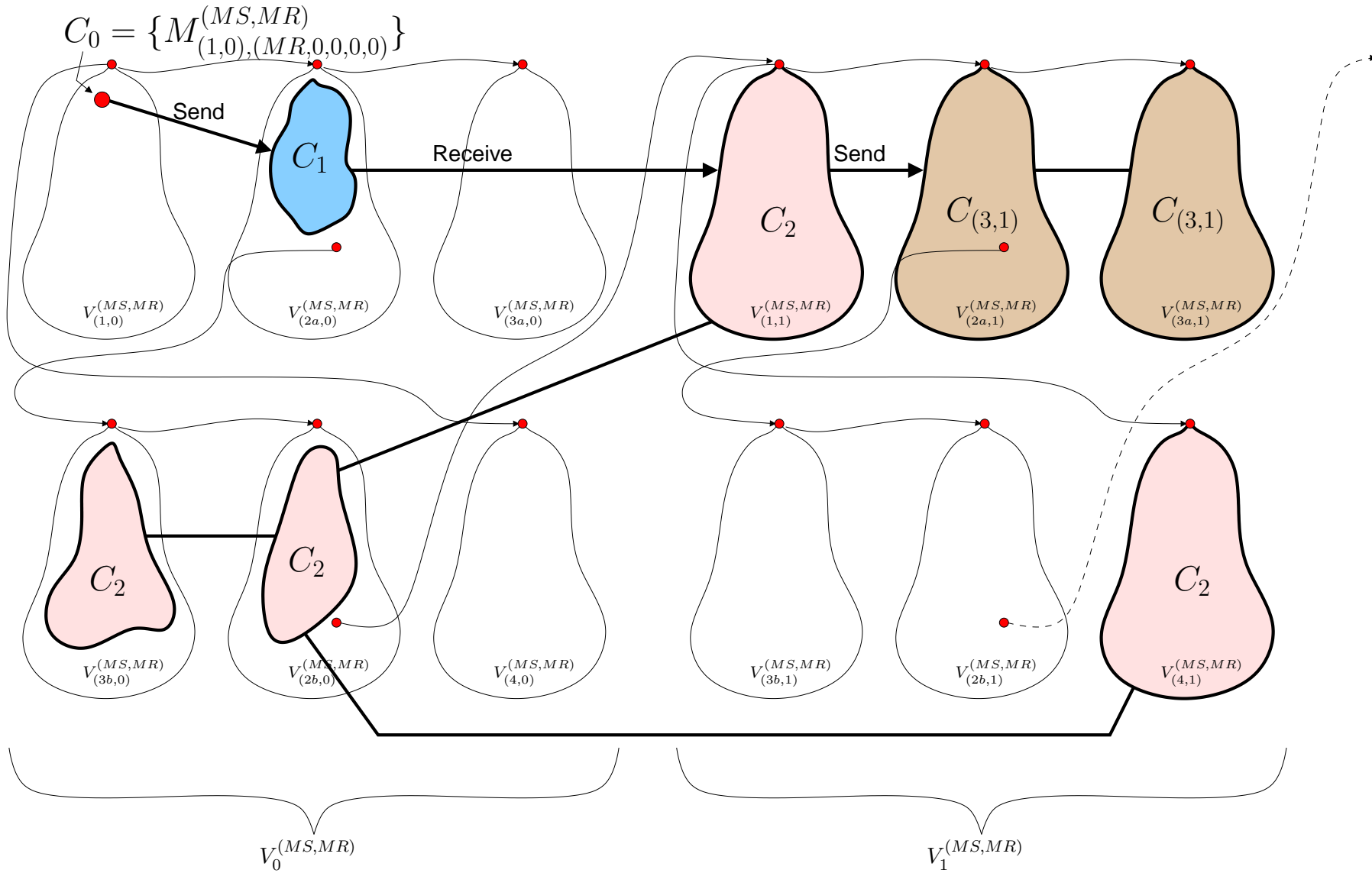


Figure 6.8: Construction of a deterministic FSA showing the addition of $C_{(3,1)}$, the successor of C_2 from Fig. 6.7.

According to the procedure in Section 2.4, the successor of $C_{(3,i)}$ is the union of the ϵ -closures of all markings in $V_{(l,i)}$. However, we again use the same approach as previously and calculate the ϵ -closure of $M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)} \in V_{(l,i)}$ and demonstrate that this equals $CLOSURE(V_{(l,i)})$. The marking we have chosen is $M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}$, as it contains the most old acknowledgements, least new messages and least retransmissions of any marking in $V_{(l,i)}$, hence intuition suggests that its ϵ -closure will span more markings (successively enable more ϵ transitions) than any other marking in $V_{(l,i)}$.

The ϵ -closure of $M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}$ is given by evaluating Equation (6.36) for $M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}$, resulting in a set of markings (specified in terms of i) which we denote $C_{(4,i)}$:

$$\begin{aligned}
 C_{(4,i)} &= Closure(M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}) \\
 &= \{M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \\
 &\cup \{M_{(2b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret + 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \\
 &\cup \{M_{(4,i \oplus_{MS} 1),(mn,an,0,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret - 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR, MR > 0\} \\
 &\cup \{M_{(1,i \oplus_{MS} 1),(mn,an,0,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \quad (6.51)
 \end{aligned}$$

Simplifying Equation (6.51) to eliminate redundant inequalities gives:

$$\begin{aligned}
 C_{(4,i)} &= \{M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR\} \\
 &\cup \{M_{(2b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret + 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \\
 &\cup \{M_{(4,i \oplus_{MS} 1),(mn,an,0,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq MR - 1, MR > 0\} \\
 &\cup \{M_{(1,i \oplus_{MS} 1),(mn,an,0,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq MR\} \quad (6.52)
 \end{aligned}$$

By inspection, this ϵ -closure covers every class 2b and 3b marking (for sender sequence number = i) and every class 1 and 4 marking (for sender sequence number = $i \oplus_{MS} 1$) defined in rows 1, 3, 5 and 6 of Table 5.3, i.e.

$$C_{(4,i)} = V_{(2b,i)}^{(MS,MR)} \cup V_{(3b,i)}^{(MS,MR)} \cup V_{(1,i \oplus_{MS} 1)}^{(MS,MR)} \cup V_{(4,i \oplus_{MS} 1)}^{(MS,MR)}$$

Lemma 8. $C_{(4,i)} = CLOSURE(V_{(l,i)})$.

Proof. From Tables 5.4, 5.6, 5.8 and 5.9, the destination marking of every outgoing ϵ -labelled arc with its source marking in $C_{(4,i)}$ is also in $C_{(4,i)}$, i.e. $CLOSURE(C_{(4,i)}) = C_{(4,i)}$. The only outgoing arcs of states in $C_{(4,i)}$ that have destinations that are not in $C_{(4,i)}$ are labelled by the **Send** service primitive, specifically, the arcs defined in row 1 of Tables 5.4 and 5.9. They correspond to the occurrence of the **send_mess** transition.

Hence, because $V_{(l,i)}$ contains only class 2b and 3b markings (with sender sequence number = i) and class 1 and 4 markings (with sender sequence number = $i \oplus_{MS} 1$), then the closure of all markings in $V_{(l,i)}$, $CLOSURE(V_{(l,i)})$, will contain only class 2b and 3b markings (with sender sequence number = i) and class 1 and 4 markings (with sender sequence number = $i \oplus_{MS} 1$). Because $C_{(4,i)}$ is the ϵ -closure of $M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)} \in V_{(l,i)}$ and it contains all class 2b, 3b, 1 and 4 markings as described, the lemma is proved. \square

Corollary 5. From Lemma 8 when $i = 1$, $C_{(4,1)} \in S^{det}$ and hence $(C_{(3,1)}, Receive, C_{(4,1)}) \in \Delta^{det}$.

This is illustrated in Fig. 6.9. Unlike C_2 , $C_{(4,1)}$ covers all class 2b and class 3b markings in $V_1^{(MS,MR)}$, whereas C_2 only covers some of the class 2b and class 3b markings in $V_0^{(MS,MR)}$. (However, both C_2 and $C_{(4,1)}$ cover all of the class 1 and class 4 markings, in $V_1^{(MS,MR)}$ and $V_2^{(MS,MR)}$ respectively.) It is important

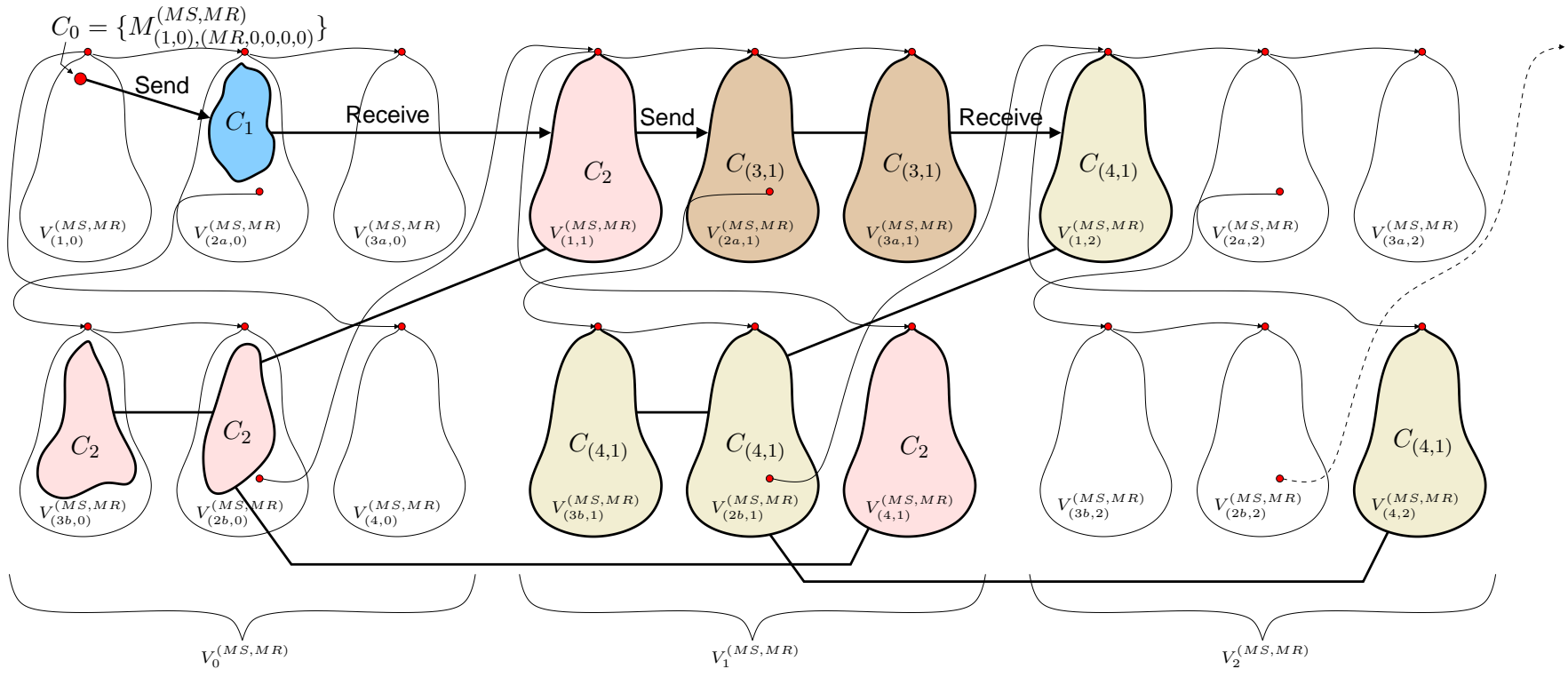


Figure 6.9: Construction of a deterministic FSA showing the addition of $C_{(4,1)}$, the successor of $C_{(3,1)}$ from Fig. 6.8.

to note, for when we construct our parametric deterministic FSA, that $C_{(4,0)}$ (obtained substituting $i = 0$ into Equation (6.52)) and C_2 are not equal.

As mentioned in the proof of Lemma 8, the only outgoing non- ϵ edges from states in $C_{(4,i)}$ are from the class 1 and 4 markings, labelled by **Send**, and corresponding to row 1 of both Table 5.4 and 5.9. $OutEdges(Closure(M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}))$ gives exactly these edges, where in fact any marking $M_{(3b,i),(0,MR,mn,0,ret)}^{(MS,MR)} \in C_{(4,i)}$ could have been used, provided $mn = ret$, for the same reasons as previously described. The successor states of these edges, from $OutEdges(Closure(M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}))$, (see Equation (6.38)) are given by $V_{(p,i)}$:

$$\begin{aligned} V_{(p,i)} &= \{s' \mid (s, \mathbf{Send}, s') \in OutEdges(Closure(M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}))\} \\ &= \{M_{(3a,i \oplus_{MS} 1),(mn,an,1,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret - 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \\ &\cup \{M_{(2a,i \oplus_{MS} 1),(mn,an,1,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret, 0 \leq mn \leq ret, 0 \leq ret \leq MR\} \end{aligned} \quad (6.53)$$

Simplifying Equation (6.53) to remove redundant inequalities gives:

$$\begin{aligned} V_{(p,i)} &= \{M_{(3a,i \oplus_{MS} 1),(mn,an,1,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq MR - 1, MR > 0\} \\ &\cup \{M_{(2a,i \oplus_{MS} 1),(mn,an,1,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq MR\} \end{aligned} \quad (6.54)$$

According to the procedure in Section 2.4, the successor of $C_{(4,i)}$ is the union of the ϵ -closures of all markings in $V_{(p,i)}$. However, as previously, we carefully select one marking in $V_{(p,i)}$ and demonstrate that its ϵ -closure is equal to $CLOSURE(V_{(p,i)})$.

The marking we have chosen is $M_{(2a,i \oplus_{MS} 1),(MR,0,1,0,0)}^{(MS,MR)}$. The motivation for choosing this particular marking is similar to the previous cases: we select a marking that will give us the largest ϵ -closure out of all the markings in $V_{(p,i)}$. The ϵ -closure of $M_{(2a,i \oplus_{MS} 1),(MR,0,1,0,0)}^{(MS,MR)}$ is given by evaluating Equation (6.15) for $M_{(2a,i \oplus_{MS} 1),(MR,0,1,0,0)}^{(MS,MR)}$, resulting in a set of markings (specified in terms of i) which we denote $C_{(5,i)}$ (after dropping the primes):

$$\begin{aligned} C_{(5,i)} &= Closure(M_{(2a,i \oplus_{MS} 1),(MR,0,1,0,0)}^{(MS,MR)}) \\ &= \{M_{(2a,i \oplus_{MS} 1),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mo \leq MR, 0 \leq mn \leq 1 + ret, \\ &\quad 0 \leq ret \leq MR\} \\ &\cup \{M_{(3a,i \oplus_{MS} 1),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mo \leq MR - 1, 0 \leq mn \leq ret + 1, \\ &\quad 0 \leq ret \leq MR, MR > 0\} \end{aligned} \quad (6.55)$$

Simplifying Equation (6.55) to eliminate redundant inequalities gives:

$$\begin{aligned} C_{(5,i)} &= \{M_{(2a,i \oplus_{MS} 1),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq 1 + ret, 0 \leq ret \leq MR\} \\ &\cup \{M_{(3a,i \oplus_{MS} 1),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq 1 + ret, 0 \leq ret \leq MR, \\ &\quad MR > 0\} \end{aligned} \quad (6.56)$$

and when replacing i with $i \oplus_{MS} 1$ in Table 5.3 we have

$$C_{(5,i)} = V_{(2a,i \oplus_{MS} 1)}^{(MS,MR)} \cup V_{(3a,i \oplus_{MS} 1)}^{(MS,MR)} \quad (6.57)$$

Lemma 9. $C_{(5,i)} = CLOSURE(V_{(p,i)})$.

Proof. Because $C_{(5,i)}$ includes all possible class 2a and 3a markings defined by Table 5.3 (with sender sequence number = $i \oplus_{MS} 1$) and from Tables 5.5 and 5.7 and mapping *Prim*, the only outgoing arcs with destination states not in $V_{(2a,i \oplus_{MS} 1)}^{(MS,MR)} \cup V_{(3a,i \oplus_{MS} 1)}^{(MS,MR)}$ are labelled by the **Receive** service primitive.

Hence, because $V_{(p,i)}$ contains only class 2a and 3a markings, then the closure of all markings in $V_{(p,i)}$, $CLOSURE(V_{(p,i)})$, will contain only class 2a and 3a markings with the same sender sequence number of $i \oplus_{MS} 1$. Because $C_{(5,i)}$ is the ϵ -closure of $M_{(2a,i \oplus_{MS} 1), (MR,0,1,0,0)}^{(MS,MR)} \in V_{(p,i)}$ and $C_{(5,i)}$ contains all class 2a and 3a markings with a sender sequence number of $i \oplus_{MS} 1$, the lemma is proved. \square

Corollary 6. From Lemma 9 when $i = 1$, $C_{(5,1)} \in S^{det}$ and hence $(C_{(4,1)}, \mathbf{Send}, C_{(5,1)}) \in \Delta^{det}$.

This result is illustrated in Fig. 6.10 for $C_{(5,1)}$. Note that $C_{(5,1)}$ covers all class 2a and class 3a markings in $V_2^{(MS,MR)}$, and that $C_{(3,1)}$ covers all class 2a and class 3a markings in $V_1^{(MS,MR)}$. By inspection of Equations (6.48) and (6.56), we find that $C_{(5,i)}$ is equal to $C_{(3,i \oplus_{MS} 1)}$. We can now state and prove a lemma that builds the rest of the structure of our parametric deterministic FSA through lazy evaluation.

Lemma 10. $\forall i \in \{0, 1, \dots, MS\}$, $C_{(3,i)} \in S^{det}$, $C_{(4,i)} \in S^{det}$, $(C_{(3,i)}, \mathbf{Receive}, C_{(4,i)}) \in \Delta^{det}$, and $(C_{(4,i)}, \mathbf{Send}, C_{(3,i \oplus_{MS} 1)}) \in \Delta^{det}$.

Proof. We know from direct construction and Corollary 3 that $C_2 \in S^{det}$ and from Corollary 4 that when $i = 1$ is substituted into Equation (6.48), we get $C_{(3,1)} \in S^{det}$ and $(C_2, \mathbf{Send}, C_{(3,1)}) \in \Delta^{det}$. From Corollary 5, we know that when substituting $i = 1$ into Equation (6.52), we get $C_{(4,1)} \in S^{det}$ and $(C_{(3,1)}, \mathbf{Receive}, C_{(4,1)}) \in \Delta^{det}$. We know that when substituting $i = 1$ into Equation (6.56) and $i = 2$ into Equation (6.48) we get $C_{(5,1)} = C_{(3,2)}$. Hence, from Corollary 6, $C_{(3,2)} \in S^{det}$ and $(C_{(4,1)}, \mathbf{Send}, C_{(3,2)}) \in \Delta^{det}$.

Repeating the application of Corollaries 5 and 6 for $i = 2, 3, \dots, MS$ we get $C_{(3,i)}, C_{(4,i)} \in S^{det}$ and $\{(C_{(3,i)}, \mathbf{Receive}, C_{(4,i)}), (C_{(4,i)}, \mathbf{Send}, C_{(3,i \oplus_{MS} 1)})\} \subset \Delta^{det}$.

The boundary cases where $i = MS$ and $i = 0$ are worthy of closer attention. $C_{(4,MS)}$ contains the initial marking, but is a different set of states than C_0 , which only contains M_0 . The successor of $C_{(4,MS)}$, upon the occurrence of the **Send** service primitive, is the set, $C_{(3,0)}$. Hence, $C_{(3,0)} \in S^{det}$. When $MR = 0$, $C_{(3,0)} = C_1$. When $MR > 0$, $C_{(3,0)} \supset C_1$ and is a distinct set of states from C_1 and hence is a distinct state in S^{det} .

From Lemma 8 and Corollary 5, when $i = 0$, we have $C_{(4,0)} \in S^{det}$ and $(C_{(3,0)}, \mathbf{Receive}, C_{(4,0)}) \in \Delta^{det}$. The set of states, $C_{(4,0)}$, is equal to C_2 when $MR = 0$. When $MR > 0$, $C_{(4,0)} \supset C_2$ and is thus a distinct state in S^{det} . We know that when substituting $i = 0$ into Equation (6.57) and $i = 1$ into Equation (6.49) we get $C_{(5,0)} = C_{(3,1)}$. Hence, from Corollary 6, $(C_{(4,0)}, \mathbf{Send}, C_{(3,1)}) \in \Delta^{det}$. Thus the lemma is proved. \square

Lemma 10, along with C_0 and $C_1 \in S^{det}$ and $(C_0, \mathbf{Send}, C_1), (C_1, \mathbf{Receive}, C_2) \in \Delta^{det}$, means we have explored all states in S^{det} and all outgoing arcs of states in S^{det} , starting from the initial state, using lazy evaluation. All that remains to complete $DFSA_{RG(MS,MR)}$ is designation of halt states.

Halt states of $DFSA_{RG(MS,MR)}$ are those subsets of states of $FSA_{RG(MS,MR)}$ that contain halt states of $FSA_{RG(MS,MR)}$. From Definition 35 the halt states of $FSA_{RG(MS,MR)}$ were defined as $\{M_{(1,i),(0,0,0,0,0)}^{(MS,MR)}, M_{(2a,i),(0,0,0,0,MR)}^{(MS,MR)}, M_{(2b,i),(0,0,0,0,MR)}^{(MS,MR)} \mid 0 \leq i \leq MS\}$. s_0^{det} is trivially a halt state. C_1 (see Equation (6.41)) is a halt state as it contains $M_{(2a,0),(0,0,0,0,MR)}^{(MS,MR)}$. C_2 (see Equation (6.43)) is a halt state as it contains $M_{(2b,0),(0,0,0,0,MR)}^{(MS,MR)}$. $C_{(3,i)}$ (Equation (6.48)) is a halt state for each $i \in \{0, 1, \dots, MS\}$ because it contains $M_{(2a,i),(0,0,0,0,MR)}^{(MS,MR)}$. Finally, $C_{(4,i)}$ (Equation (6.52)) is a halt state for each $i \in \{0, 1, \dots, MS\}$ because it contains $M_{(2b,i),(0,0,0,0,MR)}^{(MS,MR)}$. Thus all states of $DFSA_{RG(MS,MR)}$ are halt states.

Our parametric deterministic FSA, $DFSA_{RG(MS,MR)}$, is thus given by

$$DFSA_{RG(MS,MR)} = (S_{(MS,MR)}^{det}, SP, \Delta_{(MS,MR)}^{det}, s_0^{det}, F_{(MS,MR)}^{det})$$

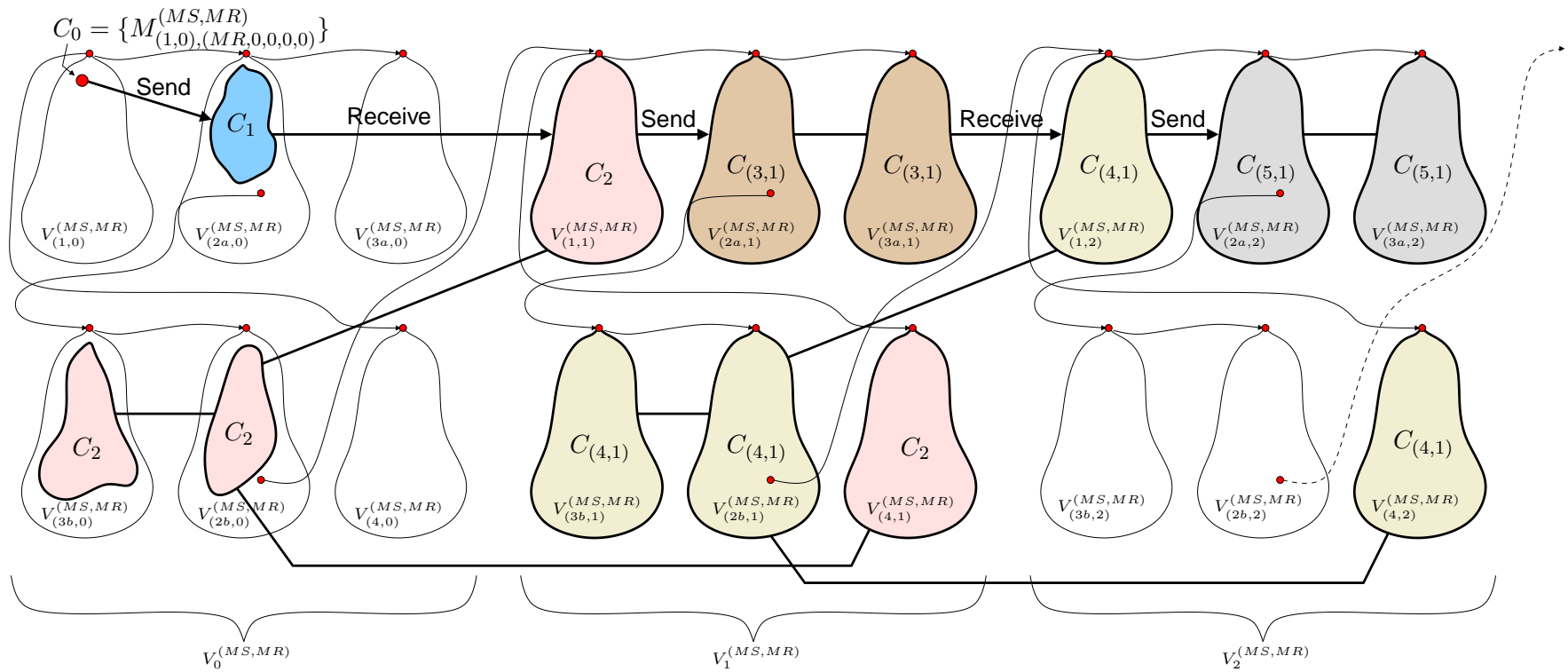


Figure 6.10: Construction of a deterministic FSA showing the addition of $C_{(5,1)}$, the successor of $C_{(4,1)}$ from Fig. 6.9.

Table 6.1: $DFSA_{RG(MS,MR)}$, where rows 4 and 5 are evaluated for $0 \leq i \leq MS$.

Source node	Arc Label	Dest. node	Dest. = Halt?
C_0	Send	C_1	true
C_1	Receive	C_2	true
C_2	Send	$C_{(3,1)}$	true
$C_{(3,i)}$	Receive	$C_{(4,i)}$	true
$C_{(4,i)}$	Send	$C_{(3,i \oplus MS1)}$	true

where:

- $S_{(MS,MR)}^{det} = \{C_0, C_1, C_2\} \cup \{C_{(3,i)}, C_{(4,i)} \mid 0 \leq i \leq MS\}$;
- $\Delta_{(MS,MR)}^{det} = \{(C_0, \mathbf{Send}, C_1), (C_1, \mathbf{Receive}, C_2), (C_2, \mathbf{Send}, C_{(3,1)})\} \cup \{(C_{(3,i)}, \mathbf{Receive}, C_{(4,i)}), (C_{(4,i)}, \mathbf{Send}, C_{(3,i \oplus MS1)}) \mid 0 \leq i \leq MS\}$;
- $s_0^{det} = \text{Closure}(M_{(1,0),(0,0,0,0)}^{(MS,MR)}) = C_0$; and
- $F_{(MS,MR)}^{det} = S_{(MS,MR)}^{det}$.

$DFSA_{RG(MS,MR)}$ is represented in tabular form in Table 6.1 and graphically in Fig. 6.11. By convention, the initial state is shown highlighted in bold and the halt states (all states) are shown as double circles in Fig. 6.11. The main loop in the lower half of the figure illustrates the repeated behaviour over all values of i , $0 \leq i \leq MS$, of alternating **Send** and **Receive** events, moving from $C_{(3,i)}$ to $C_{(4,i)}$ on a **Receive** event and from $C_{(4,i)}$ to $C_{(3,i \oplus MS1)}$ on a **Send** event.

6.5 Minimisation and Conformance to the SWP Service Language

Determinisation has removed the effect of the **MaxRetrans** parameter on the parametric FSA. However, the deterministic FSA representing the protocol language is not minimal. This is evident from the example in Fig. 6.12 for $MS = 2$ and arbitrary MR , which represents the language generated by the regular expression $(\mathbf{Send}, \mathbf{Receive})^* \mathbf{Send}^\dagger$, but which could be represented by a FSA with fewer states.

Following the minimisation procedure described in Section 2.4, from $DFSA_{RG(MS,MR)}$ (and Table 6.1) it can be seen that all states are halt states, so we begin with all states placed in the same subset, i.e. $\{C_0, C_1, C_2, C_{(3,i)}, C_{(4,i)} \mid 0 \leq i \leq MS\}$. States are now divided based on the input symbols they accept, either **Send** or **Receive**, giving us the subset $\{C_0, C_2, C_{(4,i)} \mid 0 \leq i \leq MS\}$ of states accepting the input symbol **Send**, and the subset $\{C_1, C_{(3,i)} \mid 0 \leq i \leq MS\}$ of states accepting the input symbol **Receive**. These subsets cannot be further divided, as all states in the first subset accept only a **Send**, leading to a state from the second subset, and all states in the second subset accept only a **Receive**, leading to states in the first subset. We choose the representative '1' to represent the first subset in the minimal FSA and the representative '2' to represent the second subset. Both are halt states and '1' is the initial state, as the first subset contains the initial state of the deterministic FSA, C_0 . **Send** and **Receive** edges are defined accordingly. The resulting minimised deterministic FSA is given by $MFSA_{RG(MS,MR)} = (S_{(MS,MR)}^{min}, SP, \Delta_{(MS,MR)}^{min}, 1, F_{(MS,MR)}^{min})$ where:

- $S_{(MS,MR)}^{min} = \{1, 2\}$;
- $\Delta_{(MS,MR)}^{min} = \{(1, \mathbf{Send}, 2), (2, \mathbf{Receive}, 1)\}$; and
- $F_{(MS,MR)}^{min} = S_{(MS,MR)}^{min}$.

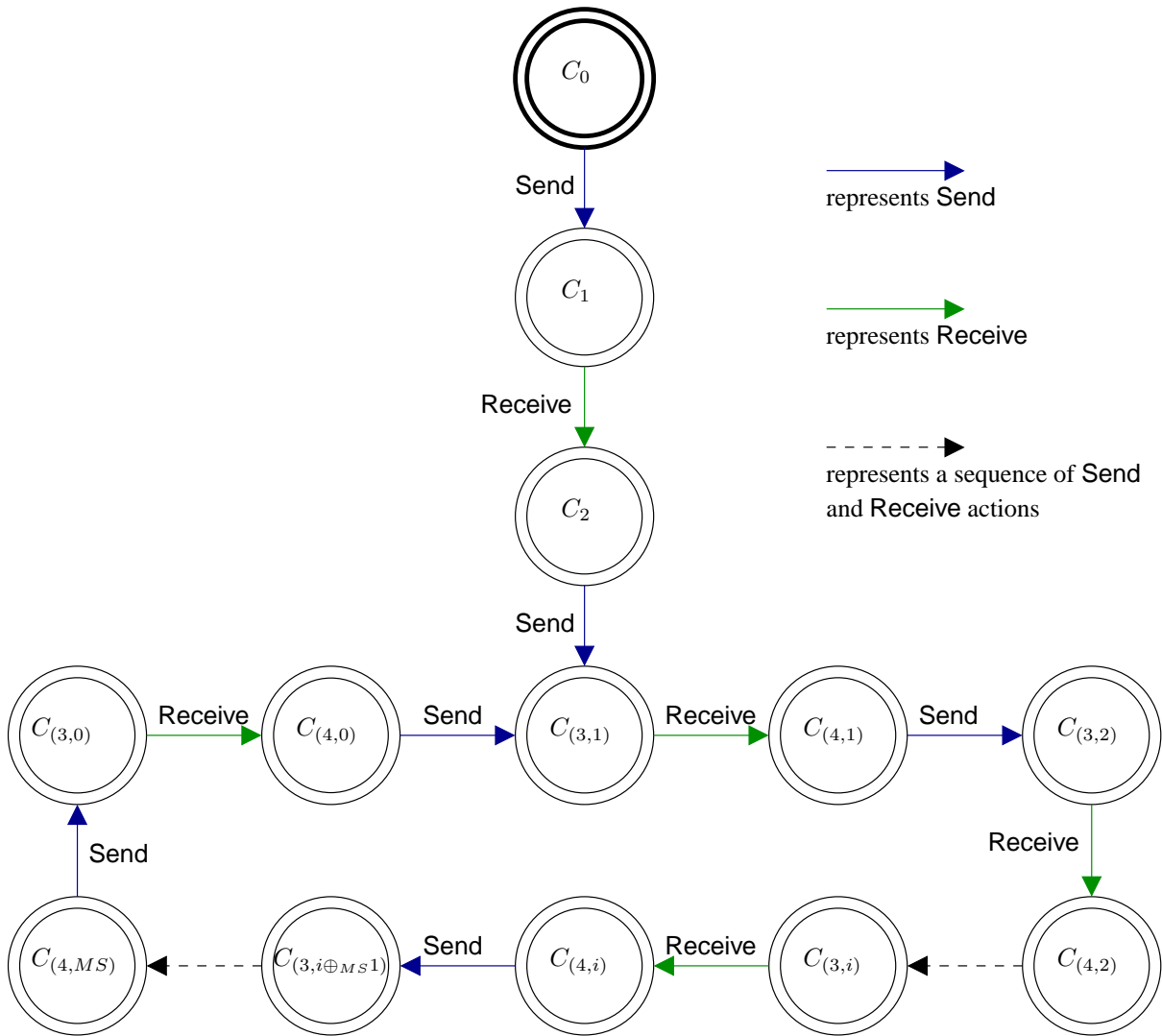


Figure 6.11: An abstract visualisation of the parametric deterministic FSA, $DFSA_{RG(MS,MR)}$.

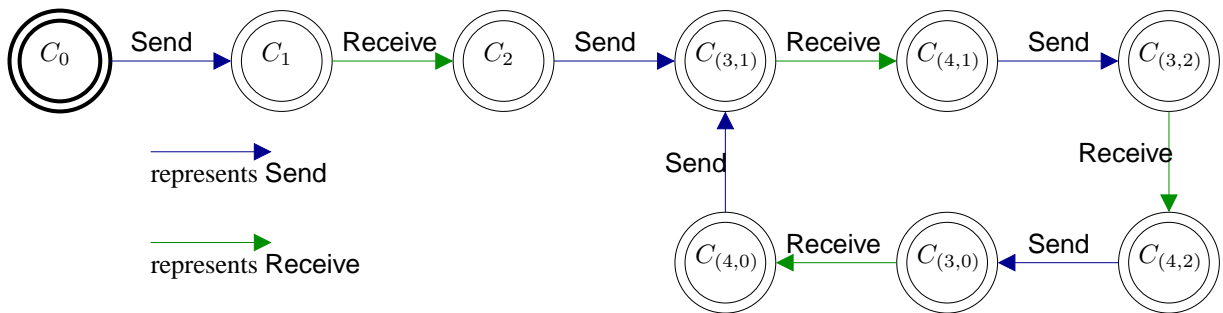


Figure 6.12: An example of the deterministic FSA for $MaxSeqNo=2$. The $MaxRetrans$ parameter has no effect on the deterministic FSA.

Table 6.2: The minimised deterministic FSA, $MFSAR_{RG(MS,MR)}$ representing the protocol language of $CPN_{(MS,MR)}$.

Source node	Arc Label	Dest. node	Dest. = Halt?
1	Send	2	true
2	Receive	1	true

$MFSAR_{RG(MS,MR)}$ is given in tabular form in Table 6.2 and is identical to the Stop-and-Wait Service Language FSA shown in Fig. 6.1. Note that this FSA is completely independent of the values of the parameters MS and MR . In the same way that determinisation removed the effect of the parameter MR , minimisation has removed the effect of the parameter MS . $MFSAR_{RG(MS,MR)}$ thus represents the protocol language for *all* of the members of the infinite family of Stop-and-Wait protocol models, i.e. *all* positive values of MS and *all* non-negative values of MR . $MFSAR_{RG(MS,MR)}$ is identical to the Stop-and-Wait service of $(\text{Send, Receive})^* \text{Send}^\dagger$. This verifies that the SWP does indeed satisfy the Stop-and-Wait property for all allowable values of the parameters, and thus Theorem 2 is proved.

Chapter 7

Conclusions and Future Work

This report documents the research activities carried out as an extension to the work in [36] and a continuation of the work in [39]. It represents completion of some of the future work activities identified in [36, 39]. This report is a further step in the protocol verification methodology for the parametric verification of the infinite class of Stop-and-Wait Protocols, that of language analysis. This work, incorporating aspects from all three reports ([36, 39] and this report) has been written up in [35].

In [39] the algebraic representation of the infinite number of RGs of a parameterised CPN model of the Stop-and-Wait Protocol was extended to include the maximum number of retransmissions parameter, in addition to the maximum sequence number parameter. This has represented a significant increase in the complexity of the algebraic expressions, as can be gauged from the quartic increase in the number of nodes and arcs of the RG in the `MaxRetrans` parameter (documented in [39]), in addition to the (only) linear increase in the `MaxSeqNo` parameter.

The algebraic expressions representing the RGs of the infinite class of Stop-and-Wait Protocols has been proved correct in [35, 39], and from these expressions a number of properties were proved. These included: an expression for the size of the RG in both parameters; the absence of unexpected deadlock; the absence of livelock; absence of unexpected dead transitions; and channel bounds. These properties were proved directly from the algebraic expressions, and thus are proved for *all* values of the (unbounded) parameters. However, the language analysis conducted in [36] for expressions in the `MaxSeqNo` parameter only (with `MaxRetrans=0`) had not been extended to take both parameters into account.

In this report, we have successfully verified that all instantiations of the Stop-and-Wait Protocol conform to the stop-and-wait service language. We have derived a parametric FSA directly from the parametric reachability graph, and performed both determinisation and minimisation directly on the parametric FSA. The parametric FSA, when determinised and minimised, reduces to a simple, non-parametric FSA describing exactly the service language. Hence, our parametric analysis of the Stop-and-Wait Protocol, parameterised with the maximum sequence number and maximum number of retransmissions, is now complete.

Outstanding future work activities also include the extension of this work to the SWP operating over a reordering medium, rather than a FIFO medium, and the extension of this work to include windows (i.e. the Sliding Window Protocol) as a step toward more complex flow control protocol such as TCP's data transfer protocol. In addition, we hope to extend the work on SWP to incorporate data independence principles [68, 85], so that properties about data (in addition to properties about sequences of events only) can be verified.

Ultimately, it is desired that the experience gained in doing so would lead to computer support tools that would partially or fully automate this procedure.

References

- [1] P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic Verification of Lossy Channel Systems: Application to the Bounded Retransmission Protocol. In *Proceedings of TACAS'99*, volume 1579 of *Lecture Notes in Computer Science*, pages 208–222. Springer-Verlag, 1999.
- [2] P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-Fly Analysis of Systems with Unbounded, Lossy, FIFO Channels. In *Proceedings of the 10th International Conference on Computer Aided Verification (CAV'98)*, Vancouver, BC, Canada, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318. Springer-Verlag, 1998.
- [3] P. Aziz Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using Forward Reachability Analysis for Verification of Lossy Channel Systems. *Formal Methods in System Design*, 25(1):39–65, 2004.
- [4] P. Aziz Abdulla and B. Jonsson. Verifying Programs with Unreliable Channels. *Information and Computation*, 127(2):91–101, June 1996.
- [5] Y. Afek and G.M. Brown. Self-Stabilization of the Alternating Bit Protocol. In *Proceedings of the 8th Symposium on Reliable Distributed Systems*, pages 80–83. IEEE Comput. Soc. Press, 1989.
- [6] A. Annichini, A. Bouajjani, and M. Sighireanu. TREX: A Tool for Reachability Analysis of Complex Systems. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, Paris, France, volume 2102 of *Lecture Notes in Computer Science*, pages 368–372. Springer-Verlag, 2001.
- [7] S. Bardin, A. Finkel, and J. Leroux. FASTer Acceleration of Counter Automata in Practice. In *Proceedings of TACAS'2004*, volume 2988 of *Lecture Notes in Computer Science*, pages 576–590. Springer, 2004.
- [8] W.A. Barrett, R. M. Bates, D. A. Gustafson, and J.D. Couch. *Compiler Construction: Theory and Practice*. Science Research Associates, 2nd edition, 1986.
- [9] K.A. Bartlett, R.A. Scantlebury, and P.T. Wilkinson. A Note on Reliable Full-Duplex Transmission over Half-Duplex Links. *Communications of the ACM*, 12(5):260–261, May 1969.
- [10] D. Basin and N. Klarlund. Automata Based Symbolic Reasoning in Hardware Verification. *Formal Methods in System Design*, 13(3):255–288, 1998.
- [11] J. Billington. Formal specification of protocols: Protocol Engineering. In *Encyclopedia of Microcomputers*, volume 7, pages 299–314. Marcel Dekker, New York, 1991.
- [12] J. Billington. Discrete Event Systems Theory and its Application to Command and Control. Technical Report CSEC-14, Computer Systems Engineering Centre Report Series, University of South Australia, December 2002.

- [13] J. Billington and G. E. Gallasch. How Stop and Wait Protocols Can Fail Over The Internet. In *Proceedings of FORTE'03*, volume 2767 of *Lecture Notes in Computer Science*, pages 209–223. Springer-Verlag, 2003. (invited paper).
- [14] J. Billington and G. E. Gallasch. An Investigation of the Properties of Stop-and-Wait Protocols over Channels which can Re-order messages. Technical Report CSEC-15, Computer Systems Engineering Centre Report Series, University of South Australia, May 2004.
- [15] J. Billington, G. E. Gallasch, and B. Han. A Coloured Petri Net Approach to Protocol Verification. In *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 210–290. Springer-Verlag, 2004.
- [16] J. Billington, G. E. Gallasch, L. M. Kristensen, and T. Mailund. Exploiting equivalence reduction and the sweep-line method for detecting terminal states. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 34(1):23–37, January 2004.
- [17] J. Billington, G. E. Gallasch, and L. Petrucci. FAST Verification of the Class of Stop-and-Wait Protocols modelled by Coloured Petri Nets. *Nordic Journal of Computing*, Vol. 12(3):251–274, 2005.
- [18] J. Billington, G. E. Gallasch, and L. Petrucci. Transforming Coloured Petri Nets to Counter Systems for Parametric Verification: A Stop-and-Wait Protocol Case Study. In *Proceedings of 2nd International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES'05)*, Rennes, France, TUCS General Publication, No. 39, pages 37–55, May 2005.
- [19] J. Billington and B. Han. Closed Form Expressions for the State Space of TCP's Data Transfer Service Operating over Unbounded Channels. In *Proceedings of 27th Australasian Computer Science Conference (ACSC'2004)*, Dunedin, New Zealand, volume 26 of *Conferences in Research and Practice in Information Technology*, pages 31–39. Australian Computer Society, January 2004.
- [20] J. Billington, G.R. Wheeler, and M.C. Wilbur-Ham. PROTEAN: A High-level Petri Net Tool for the Specification and Verification of Communication Protocols. *IEEE Transactions on Software Engineering*, 14(3):301–316, March 1988.
- [21] J. Billington, M.C. Wilbur-Ham, and M.Y. Bearman. Automated Protocol Verification. In *Protocol Specification, Testing and Verification, V*, pages 59–70. North Holland, Amsterdam, 1986.
- [22] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Comput. Networks and ISDN Sys.*, 14(1):25–59, 1987.
- [23] M. Bozzano and G. Delzanno. Beyond Parameterized Verification. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 221–235. Springer-Verlag, 2002.
- [24] S. Budkowski and P. Dembinski. An Introduction to Estelle: A Specification Language for Distributed Systems. *Comput. Networks and ISDN Sys.*, 14(1):3–23, 1987.
- [25] CADP homepage. <http://www.inrialpes.fr/vasy/cadp/>.
- [26] CCITT. ISDN user-network interface Data link layer specification. Technical report, Draft Recommendation Q.921, Working Party XI/6, Issue 7, Jan. 1984.
- [27] S. Christensen, L. M. Kristensen, and T. Mailund. A Sweep-Line Method for State Space Exploration. In *Proceedings of TACAS 2001*, volume 2031 of *Lecture Notes in Computer Science*, pages 450–464. Springer-Verlag, 2001.

- [28] E.M. Clarke and J. M. Wing. Formal Methods: State of the Art and Future Directions. *ACM Computing Surveys*, 28(4):pages 626–643, December 1996.
- [29] Design/CPN Online. <http://www.daimi.au.dk/designCPN/>.
- [30] M. Diaz. Modelling and Analysis of Communication and Co-operation Protocols Using Petri Net Based Models. In *Protocol Specification, Testing and Verification*, pages 465–510. North-Holland, 1982.
- [31] E.A. Emerson, S. Jha, and D. Peled. Combining Partial Order and Symmetry Reduction. In *Proceedings of TACAS'97*, volume 1217 of *Lecture Notes in Computer Science*, pages 35–49. Springer-Verlag, 1997.
- [32] FAST - Fast Acceleration of Symbolic Transition systems.
<http://www.lsv.ens-cachan.fr/fast/>.
- [33] A. Finkel and J. Leroux. How to Compose Presburger-Accelerations: Applications to Broadcast Protocols. In *Proceedings of FST&TCS'2002*, volume 2556 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2002.
- [34] FSM Library, AT&T Research Labs.
<http://www.research.att.com/sw/tools/fsm/>.
- [35] G. E. Gallasch. *Parametric Verification of the Class of Stop-and-Wait Protocols*. PhD thesis, Computer Systems Engineering Centre, School of Electrical and Information Engineering, University of South Australia, Adelaide, Australia, May 2007.
- [36] G. E. Gallasch and J. Billington. Towards the Parametric Verification of the Class of Stop-and-Wait Protocols over Ordered Channels. Technical Report CSEC-21, Computer Systems Engineering Centre Report Series, University of South Australia, March 2005, revised June 2005.
- [37] G. E. Gallasch and J. Billington. Using Parametric Automata for the Verification of the Stop-and-Wait Class of Protocols. In *Proceedings of ATVA 2005*, volume 3707 of *Lecture Notes in Computer Science*, pages 457–473. Springer-Verlag, 2005.
- [38] G. E. Gallasch and J. Billington. A Parametric State Space for the Analysis of the Infinite Class of Stop-and-Wait Protocols. In *Proceedings of the 13th International SPIN Workshop on Model Checking of Software (SPIN 2006)*, volume 3925 of *Lecture Notes in Computer Science*, pages 201–218. Springer-Verlag, March-April 2006.
- [39] G. E. Gallasch and J. Billington. Parametric Verification of the Class of Stop-and-Wait Protocols over Ordered Channels. Technical Report CSEC-23, Computer Systems Engineering Centre Report Series, University of South Australia, December 2006.
- [40] B. Han. *Formal Specification of the TCP Service and Verification of TCP Connection Management*. PhD thesis, Computer Systems Engineering Centre, School of Electrical and Information Engineering, University of South Australia, Adelaide, Australia, December 2004.
- [41] J. E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2nd edition, 2001.
- [42] ISO/IEC. *Software and Systems Engineering – High-level Petri Nets – Part 1: Concepts, Definitions and Graphical Notation*. ISO/IEC 15909-1, 1 December 2004.
- [43] ITU-T. *Recommendation Z.100: Functional Specification and Description Language (SDL)*. International Telecommunications Union, 2002.

- [44] ITU-T. *Recommendation H.245, Control protocol for multimedia communication*. International Telecommunications Union, October 2005.
- [45] M. Jørgensen J. Jensen and N. Klarlund. Monadic Second-Order Logic for Parameterized Verification. Technical Report RS-94-10, Basic Research in Computer Science (BRICS), Department of Computer Science, University of Aarhus, May 1994. 14 pages.
- [46] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Vol. 1, Basic Concepts*. Monographs in Theoretical Computer Science. Springer-Verlag, 2nd edition, 1997.
- [47] K. Jensen, L. M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9(3-4):213–254, June 2007.
- [48] R. Kaivola. *Equivalences, Preorders and Compositional Verification for Linear Time Temporal Logic and Concurrent Systems*. PhD thesis, Department of Computer Science, University of Helsinki, Finland, 1996.
- [49] R. Kaivola. Using Compositional Predorders in the Verification of Sliding Window Protocol. In *Proceedings of CAV'97*, volume 1254 of *Lecture Notes in Computer Science*, pages 48–59. Springer-Verlag, 1997.
- [50] D. Kelley. *Automata and Formal Languages: An Introduction*. Prentice-Hall, 1995.
- [51] D. E. Knuth. Verification of Link-Level Protocols. *BIT*, 21:31–36, 1981.
- [52] L. M. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, 1998.
- [53] L. M. Kristensen and T. Mailund. A Generalised Sweep-Line Method for Safety Properties. In *Proceedings of FME'02*, volume 2391 of *Lecture Notes in Computer Science*, pages 549–567. Springer-Verlag, 2002.
- [54] R. P. Kurshan and K. L. McMillan. A Structural Induction Theorem for Processes. In *Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada*, pages 239–247. ACM Press, 1989.
- [55] R. P. Kurshan and K. L. McMillan. A Structural Induction Theorem for Processes. *Information and Computation*, 117(1):1–11, 1995.
- [56] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, 2nd edition, 1998.
- [57] P. Linz. *An Introduction to Formal Languages and Automata*. Jones and Bartlett Publishers, 3rd edition, 2001.
- [58] L. Liu. *Towards Parametric Verification of the Capability Exchange Signalling Protocol*. PhD thesis, Computer Systems Engineering Centre, School of Electrical and Information Engineering, University of South Australia, Adelaide, Australia, August 2006.
- [59] L. Liu and J. Billington. Tackling the Infinite State Space of a Multimedia Control Protocol Service Specification. In *Proceedings of ICATPN'02*, volume 2360 of *Lecture Notes in Computer Science*, pages 273–293. Springer-Verlag, 2002.
- [60] W.C. Lynch. Reliable Full-Duplex File Transmission over Half-Duplex Telephone Lines. *Communications of the ACM*, 11(6):407–410, June 1968.

- [61] T. Mailund. Analysing Infinite-State Systems by Combining Equivalence Reduction and the Sweep-Line Method. In *Proceedings of ICATPN'02*, volume 2360 of *Lecture Notes in Computer Science*, pages 314–334. Springer-Verlag, 2002.
- [62] K. L. McMillan and J. Schwalbe. Formal Verification of the Gigamax Cache Consistency Protocol. In *Proceedings of the International Symposium on Shared Memory Multiprocessors*, pages 242–251. Information Processing Society Japan, 1991.
- [63] M. Mohri. Generic Epsilon-Removal and Input Epsilon-Normalisation Algorithms for Weighted Transducers. *International Journal of Foundations of Computer Science*, 13(1):129–143, 2002.
- [64] D. Peled. Ten Years of Partial Order Reduction. In *Proceedings of CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 17–28. Springer-Verlag, 1998.
- [65] F. Pong and M. Dubois. Verification Techniques for Cache Coherence Protocols. *ACM Computing Surveys*, 29(1):82–126, 1997.
- [66] W. Reisig. *Elements of Distributed Algorithms: Modelling and Analysis with Petri Nets*. Springer-Verlag, 1998.
- [67] A. Roychoudhury and I. V. Ramakrishnan. Automated inductive verification of parameterized protocols. In *Proceedings of CAV 2001*, volume 2102 of *Lecture Notes in Computer Science*, pages 25–37. Springer-Verlag, 2001.
- [68] K. Sabnani. An Algorithmic Technique for Protocol Verification. *IEEE Transactions on Communications*, 36(8):924–931, 1988.
- [69] M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, 2nd edition, 2006.
- [70] W. Stallings. *Data and Computer Communications*. Prentice Hall, 7th edition, 2004.
- [71] Standard ML of New Jersey. <http://cm.bell-labs.com/cm/cs/what/smlnj/>.
- [72] L.J. Steggle and P. Kosiuczenko. A Timed Rewriting Logic Semantics for SDL: a case study of the Alternating Bit Protocol. *Electronic Notes in Theoretical Computer Science*, 15, 1998.
- [73] N. V. Stenning. A Data Transfer Protocol. *Computer Networks*, 1(2):99–110, 1976.
- [74] U. Stern and D. L. Dill. Automatic Verification of the SCI Cache Coherence Protocol. In *Proceedings of the IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, volume 987 of *Lecture Notes in Computer Science*, pages 21–34. Springer-Verlag, 1995.
- [75] I. Suzuki. Formal Analysis of the Alternating Bit Protocol by Temporal Petri Nets. *IEEE Transactions on Software Engineering*, 16(11):1273–1281, 1990.
- [76] I. Suzuki. Specification and Verification of the Alternating Bit Protocol by Temporal Petri Nets. In *Proceedings of the 32nd Midwest Symposium on Circuits and Systems*, pages 157–160. IEEE Press, 1990.
- [77] A. Tanenbaum. *Computer Networks*. Prentice Hall, 4th edition, 2003.
- [78] The TReX Tool. <http://www.liafa.jussieu.fr/~sighirea/trex/>.
- [79] K. J. Turner (Ed.). *Using Formal Description Techniques: An Introduction to Estelle, Lotos and SDL*. Wiley Series in Communication and Distributed Systems. John Wiley & Sons, 1993.
- [80] A. Valmari. Compositionality in State Space Verification Methods. In *Proceedings of ICATPN'96*, volume 1091 of *Lecture Notes in Computer Science*, pages 29–56. Springer-Verlag, 1996.

- [81] A. Valmari. The State Explosion Problem. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 429–528. Springer-Verlag, 1998.
- [82] A. Valmari and I. Kokkarinen. Unbounded Verification Results by Finite-State Compositional Techniques: 10^{any} States and Beyond. In *Proceedings of International Conference on Application of Concurrency to System Design*, pages 75–85. IEEE Computer Society, March 1998.
- [83] A. Valmari and M. Tienari. *An Improved Failures Equivalence for Finite-State Systems with a Reduction Algorithm*, volume XI of *Protocol Specification, Testing, and Verification*, pages 3–18. Elsevier Science Publishers, 1991.
- [84] G. van Noord. Treatment of Epsilon Moves in Subset Construction. *Computational Linguistics*, 26(1):61–76, 2000.
- [85] P. Wolper. Expressing Interesting Properties of Programs in Propositional Temporal Logic. In *Proceedings of the 13th Annual ACM Symposium on Principles of Programming Languages (POPL)*, pages 184–193. ACM Press, 1986.
- [86] L. Zuck and A. Pnueli. Model checking and abstraction to the aid of parameterized systems. *Computer Languages, Systems & Structures*, 30(3-4, Analysis and Verification):139–169, 2004.