Parametric Verification of the Class of Stop-and-Wait Protocols

GUY EDWARD GALLASCH
B.Eng. (Comp Sys) Hons.

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

A thesis submitted in total fulfilment
of the requirements for the degree of
Doctor of Philosophy in Computer Systems Engineering

28 May 2007
## Contents

1 Introduction  
1.1 Background and Motivation ................................................. 1  
1.2 Research Aims ................................................................. 3  
1.3 Scope ............................................................................. 4  
1.4 Overview of this Thesis ......................................................... 4  

2 Formal Techniques  
2.1 Coloured Petri nets .............................................................. 7  
2.1.1 Multisets .................................................................... 8  
2.1.2 Operations on Multisets .................................................. 9  
2.1.3 High-level Petri Net Graph ............................................. 9  
2.1.4 A Simple Coloured Petri Net Example ............................... 11  
2.1.5 The Corresponding High-level Petri Net Graph .................... 14  
2.1.6 Reachability Graphs ....................................................... 15  
2.1.7 Coloured Petri Net Tool Support ..................................... 17  
2.2 Finite State Automata .......................................................... 17  
2.2.1 Languages ................................................................ 17  
2.2.2 Formal Definitions of Finite State Automata ...................... 18  
2.2.3 The Language of a Deterministic FSA .............................. 20  
2.2.4 Automata Reduction Techniques .................................... 21  
2.2.5 FSA Tool Support ......................................................... 26  
2.3 Concluding Remarks ......................................................... 26  

3 Protocol Verification Methodology: An Overview  
3.1 Overview of the Protocol Verification Methodology ................. 27  
3.1.1 Service specification ...................................................... 27  
3.1.2 Protocol Specification .................................................... 28
3.1.3 Analysis ................................................................. 29
3.1.4 Comparison .......................................................... 29
3.2 Summary and Limitations ............................................ 30
3.3 Concluding Remarks .................................................. 31

4 Previous Work .......................................................... 32
4.1 Non-Symbolic Verification of SWP ................................. 32
4.2 Symbolic Analysis using TRex ....................................... 35
4.3 A Compositional Behavioural Fixed Point Approach .......... 36
4.4 The Sliding Window Protocol ....................................... 36
4.5 Algebraic Expressions for Other Systems ......................... 37
4.6 Our own Previous Work ............................................. 38
4.7 Concluding Remarks ................................................. 40

5 Stop-and-Wait Flow Control Protocol ............................... 41
5.1 Introduction to the Stop-and-Wait Protocol ....................... 41
  5.1.1 Sender Procedures ............................................... 43
  5.1.2 Receiver Procedures ............................................ 45
5.2 The Stop-and-Wait Service ......................................... 47
  5.2.1 Sequences of Events ............................................ 47
  5.2.2 Data Independence ............................................... 49
5.3 Modelling Scope and Assumptions ................................ 50
5.4 Parameterisation ...................................................... 51
5.5 The Parameterised Stop-and-Wait CPN Model .................... 52
  5.5.1 The Sender ...................................................... 54
  5.5.2 The Receiver .................................................... 55
  5.5.3 The Underlying Medium ....................................... 56
  5.5.4 Discussion of Modelling Decisions ............................ 58
5.6 The Parameterised Stop-and-Wait High-level Petri Net Graph 59
5.7 Properties for Verification .......................................... 64
  5.7.1 Absence of Unexpected Dead Markings ....................... 64
  5.7.2 Absence of Livelock ............................................ 65
  5.7.3 Absence of Unexpected Dead Transitions .................... 66
  5.7.4 Channel Bounds .................................................. 66
  5.7.5 Conformance to the Service Language ....................... 67
## CONTENTS

5.8 Concluding Remarks ................................................. 67

### 6 Model Behaviour ................................. 68

6.1 The SWP Without Retransmissions .......................... 68

6.1.1 Regular Patterns in the Concrete Reachability Graphs ........................................... 70

6.1.2 Parametric Reachability Graph in $\text{MaxSeqNo}$ ........................................... 73

6.2 The SWP With Retransmissions .......................... 74

6.3 Investigating Behavioural Properties ................. 76

6.3.1 Singleton Multiset Markings ......................... 76

6.3.2 Sequence Number Independent Behaviour .......... 78

6.4 Concluding Remarks .......................... 92

### 7 A Notation for Markings and Arcs ....... 93

7.1 Notational Conventions .......................... 93

7.2 Marking and Arc Notation .......................... 93

7.2.1 Marking Notation ........................................... 94

7.2.2 Arc Notation ........................................... 97

7.3 Classifying Markings ........................................... 102

7.4 Parametric Marking and Arc Notation ............ 103

7.5 Queue Lengths ........................................... 104

7.6 Downward-Closed Sets of Markings ............... 105

7.7 Concluding Remarks ........................................... 106

### 8 The Parametric SWP Reachability Graph .... 107

8.1 Sets of Markings ...................................... 107

8.2 Sets of Arcs ...................................... 108

8.3 The Parametric Reachability Graph ............... 109

8.4 A Spanning of all Markings .......................... 109

8.5 Containment of all Successors .......................... 131

8.6 Concluding Remarks ...................................... 154

### 9 Analysis of the Parametric Reachability Graph .... 155

9.1 Size of the Reachability Graph ....................... 155

9.1.1 The Number of Markings ....................... 156

9.1.2 The Number of Arcs ....................... 159

9.2 Absence of Unexpected Dead Markings ........... 166
CONTENTS

9.3 Absence of Livelock .................................................. 167
9.4 Absence of Unexpected Dead Transitions ......................... 168
9.5 Channel Bounds ...................................................... 169
9.6 Concluding Remarks ................................................. 170

10 Parametric Language Analysis .................................. 171
10.1 Obtaining the Parametric Protocol Language .................... 172
10.2 Epsilon Closures .................................................... 173
  10.2.1 The $\epsilon$-closure of Nodes in $V^{(MS,MR)}_{(2a,i)}$ ................. 174
  10.2.2 The $\epsilon$-closure of Nodes in $V^{(MS,MR)}_{(3b,i)}$ ...................... 182
10.3 Determinisation ..................................................... 191
10.4 Minimisation and Conformance to the SWP Service Language ..................... 205
10.5 Concluding Remarks ................................................. 208

11 Conclusions and Future Work .................................... 209
11.1 Contribution of this Research ..................................... 210
  11.1.1 Algebraic Representation of an Infinite Class of Reachability Graphs .......... 210
  11.1.2 Parametric Verification of Properties directly from the Algebraic Expressions ........ 210
  11.1.3 Parametric Language Analysis directly on the Parametric Reachability Graph .... 211
  11.1.4 Toward a Parametric Verification Methodology .............................. 212
11.2 Future Work .......................................................... 213
  11.2.1 Extending the Analysis to Reordering Channels .............................. 213
  11.2.2 The Importance of Data Independence ...................................... 213
  11.2.3 Using Equivalence Classes to Model In-Order Channels ...................... 214
  11.2.4 Developing and Automating a Parametric Verification Methodology .............. 215

A Concepts from Algebraic Specification .......................... 230
  A.1 Signatures .......................................................... 230
  A.2 Boolean Signature .................................................. 231
  A.3 Variables ........................................................... 231
  A.4 Terms built from a Signature and Variables .............................. 231
  A.5 Many-sorted Algebras ............................................. 232
  A.6 Assignment and Evaluation ....................................... 233

B Reachability Graphs for MaxRetrans=0 ............................ 234
  B.1 Reachability Graphs .............................................. 234
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A simple Coloured Petri Net example.</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Declarations for the simple Coloured Petri Net example in Fig. 2.1.</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>A Simple Nondeterministic FSA.</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>The language equivalent Deterministic FSA of the simple Nondeterministic FSA in Fig. 2.3.</td>
<td>23</td>
</tr>
<tr>
<td>2.5</td>
<td>Refining the partition of states based on distinguishability.</td>
<td>25</td>
</tr>
<tr>
<td>2.6</td>
<td>The language equivalent Minimised FSA of the Deterministic FSA in Fig. 2.4.</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>Protocol verification methodology</td>
<td>30</td>
</tr>
<tr>
<td>5.1</td>
<td>Key procedures of the Sender.</td>
<td>46</td>
</tr>
<tr>
<td>5.2</td>
<td>Key procedures of the Receiver.</td>
<td>46</td>
</tr>
<tr>
<td>5.3</td>
<td>An FSA of the Stop-and-Wait Service Language.</td>
<td>47</td>
</tr>
<tr>
<td>5.4</td>
<td>An FSA of an alternative Stop-and-Wait Service Language.</td>
<td>48</td>
</tr>
<tr>
<td>5.5</td>
<td>The CPN of the Stop-and-Wait Protocol operating over an in-order (FIFO) underlying medium.</td>
<td>53</td>
</tr>
<tr>
<td>5.6</td>
<td>Global Declarations for the Stop-and-Wait Protocol CPN in Fig. 5.5.</td>
<td>54</td>
</tr>
<tr>
<td>6.1</td>
<td>The RGs of $CPN_{(1,0)}$ and $CPN_{(2,0)}$: a) $RG_{(2,0)}$; b) $RG_{(2,0)}$.</td>
<td>69</td>
</tr>
<tr>
<td>6.2</td>
<td>The fragment of $RG_{(MS,0)}$ for any $i, 0 \leq i \leq MS$.</td>
<td>73</td>
</tr>
<tr>
<td>8.1</td>
<td>An illustration at an abstract level of the procedure undertaken for the proof of the Spanning Lemma, Lemma 8.1.</td>
<td>113</td>
</tr>
<tr>
<td>8.2</td>
<td>The class 1 markings from $V_i^{(MS,MR)}$, represented by the value of $mo$ and $ao$, reachable from $M_{(1,i),(MR,0,0,0,0)}^{(MS,MR)}$ through repeated successive firings of receive_mess followed by send_ack.</td>
<td>115</td>
</tr>
<tr>
<td>8.3</td>
<td>The class 1 markings from $V_i^{(MS,MR)}$, represented by the value of $mo$ and $ao$, reachable from the markings in Fig. 8.2 through repeated firing of mess_loss.</td>
<td>117</td>
</tr>
</tbody>
</table>
8.4 The class 1 markings from $V_i^{(MS,MR)}$, represented by the value of $mo$ and $ao$, reachable from the markings in Fig. 8.2 by considering all possible interleavings of mess, loss and ack loss. ........................................ 117

8.5 An abstract view of the spanning of all markings in the set $V_{i}^{(MS,MR)}(1,i)$, with an arc from $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$ to $M^{(MS,MR)}_{(2a,1),(MR,0,1,0,0)}$ for the future spanning of all markings in $V_{i}^{(MS,MR)} (2a,i)$ ........................................................................................................ 118

8.6 An abstract view of the spanning of all markings in $V_{i}^{(MS,MR)}(1,i)$ and $V_{i}^{(MS,MR)} (2a,i)$, with an arc from $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)}$ to $M^{(MS,MR)}_{(3a,i),(MR−1,0,1,0,0)}$ for the future spanning of all markings in $V_{i}^{(MS,MR)}(3a,i)$ ........................................................................................................ 119

8.7 An abstract view of the spanning of all markings in $V_{i}^{(MS,MR)}(1,i)$, $V_{i}^{(MS,MR)} (2a,i)$ and $V_{i}^{(MS,MR)} (3a,i)$, with an arc from $M^{(MS,MR)}_{(2a,i),(0,MR,1,0,0)}$ to $M^{(MS,MR)}_{(3a,i),(0,MR,0,0,0)}$ for the future spanning of all markings in $V_{i}^{(MS,MR)} (3a,i)$ ........................................................................................................ 122

8.8 An abstract view of the spanning of all markings in $V_{i}^{(MS,MR)}(1,i)$, $V_{i}^{(MS,MR)} (2a,i)$, $V_{i}^{(MS,MR)} (3a,i)$ and $V_{i}^{(MS,MR)}(2b,i)$, with an arc from $M^{(MS,MR)}_{(3a,i),(0,MR,0,0,0)}$ to $M^{(MS,MR)}_{(2b,i),(0,MR,0,1,0)}$ for the future spanning of all markings in $V_{i}^{(MS,MR)}(2b,i)$ ........................................................................................................ 124

8.9 An abstract view of the spanning of all markings in $V_{i}^{(MS,MR)}(1,i)$, $V_{i}^{(MS,MR)} (2a,i)$, $V_{i}^{(MS,MR)} (3a,i)$, $V_{i}^{(MS,MR)} (3b,i)$ and $V_{i}^{(MS,MR)}(2b,i)$, with an arc from $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$ to $M^{(MS,MR)}_{(4,i),(MR−1,0,0,0,0)}$ for the future spanning of all markings in $V_{i}^{(MS,MR)}(4,i)$ ........................................................................................................ 126

8.10 An abstract view of the complete spanning of all markings in $V_{i}^{(MS,MR)}$, i.e. all markings in $\bigcup_{class \in \{1,2a,2b,3a,3b,4\}} V_{i}^{(MS,MR)}(class,i)$ ........................................................................................................ 128

8.11 An abstract view of the complete spanning of all markings in $V_{i}^{(MS,MR)}$, i.e. all markings in $\bigcup_{class \in \{1,2a,2b,3a,3b,4\}} V_{i}^{(MS,MR)}(class,i)$ ........................................................................................................ 130

8.12 An abstract view of the complete spanning of the markings in $\bigcup_{0 \leq t \leq MS} V_{i}^{(MS,MR)}$ ........................................................................................................ 132

10.1 The markings, represented by $mo$, $ao$ and $mn$, reachable from $M_{(2a,i),(mo,ao,mn,0,ref)}^{(MS,MR)}$ through repeated successive firings of receive mess followed by send ack. ........................................ 176

10.2 The markings from $V_{a}^{(MS,MR)}(2a,i)$ and $V_{b}^{(MS,MR)}(3a,i)$, represented by $mo$, $ao$ and $mn$, reachable from the markings in Fig. 10.1 through repeated firing of timeout retrans. ........................................................................................................ 178

10.3 The coverage of $V_{a}^{(MS,MR)}$ and $V_{b}^{(MS,MR)}$ over $V_{i}^{(MS,MR)}$ ........................................................................................................ 180

10.4 The coverage of $V_{a}^{(MS,MR)}$, $V_{e}^{(MS,MR)}$, $V_{f}^{(MS,MR)}$ and $V_{g}^{(MS,MR)}$ over $V_{i}^{(MS,MR)} \cup V_{i \in MS}^{(MS,MR)}$ ........................................................................................................ 187

10.5 Construction of a deterministic FSA using subset construction with lazy evaluation, showing the initial state, $s_{0}^{det}$, and its only successor, $C_{1}$ ........................................................................................................ 193

10.6 Construction of a deterministic FSA showing the addition of $C_{2}$, the successor of $C_{1}$ from Fig. 10.5. ........................................................................................................ 195
10.7 Construction of a deterministic FSA showing the addition of $C_{(3,1)}$, the successor of $C_2$ from Fig. 10.6. ................................................................. 199
10.8 Construction of a deterministic FSA showing the addition of $C_{(1,1)}$, the successor of $C_{(3,1)}$ from Fig. 10.7. ................................................................. 201
10.9 Construction of a deterministic FSA showing the addition of $C_{(5,1)}$, the successor of $C_{(4,1)}$ from Fig. 10.8. ................................................................. 204
10.10 An abstract visualisation of the parametric deterministic FSA, $DFSA_{RG(MS,MR)}$. ................................................................. 206
10.11 An example of the deterministic FSA for MaxSeqNo=2. The MaxRetrans parameter has no effect on the deterministic FSA. ................................................................. 207

B.1 The reachability graph, $RG_{(1,0)}$, when MaxSeqNo=1 and MaxRetrans=0. ................. 235
B.2 The reachability graph, $RG_{(2,0)}$, when MaxSeqNo=2 and MaxRetrans=0. ................. 236
B.3 The reachability graph, $RG_{(3,0)}$, when MaxSeqNo=3 and MaxRetrans=0. ................. 237
B.4 The reachability graph, $RG_{(4,0)}$, when MaxSeqNo=4 and MaxRetrans=0. ................. 238
List of Tables

5.1 Alternative models of loss operating on a channel containing \([0, 0, 0, 1, 1, 1]\). ........................................... 57
6.1 \(V_{(i,i)}^{(MS,0)}\) for \(0 \leq i \leq MS\) ................................................................. 71
6.2 \(V_{(i,i@A,1)}^{(MS,0)}\) for \(0 \leq i \leq MS\) ................................................................. 71
6.3 \(A_{(i,i)}^{(MS,0)}\) for \(0 \leq i \leq MS\) ................................................................. 72
6.4 \(A_{(i,i@A,1)}^{(MS,0)}\) for \(0 \leq i \leq MS\) ................................................................. 72
6.5 RG Statistics of the CPN in Figs. 5.5 and 5.6 for various parameter values. ............... 75
6.1 \(V_{(MS;0)}(i;i)\) for \(0 \leq i \leq MS\) ................................................................. 71
6.2 \(V_{(MS;0)}(i;i@A,0)\) for \(0 \leq i \leq MS\) ................................................................. 71
6.3 \(A_{(MS;0)}(i,i)\) for \(0 \leq i \leq MS\) ................................................................. 72
6.4 \(A_{(MS;0)}(i,i@A,0)\) for \(0 \leq i \leq MS\) ................................................................. 72
7.1 A mapping from an augmented transition name, \(atn\), to its corresponding enabled binding element, given a source marking, \(M = M_{(sstate,rstate),(ssn,rsn),(ma,oa,an,ret)}^{(MS,MR)}\) that enables TransMap\((atn)\). ......................................................... 101
7.2 Classification of markings into Classes of markings based on the state of the sender and receiver. ................................................................. 103
8.1 \(V_{i}^{(MS,MR)} = \bigcup_{class \in Class} V_{(class,i)}^{(MS,MR)}\), for \(0 \leq i \leq MS\) and Class = \(\{1, 2a, 2b, 3a, 3b, 4\}\). 108
8.2 The set of arcs, \(A_{(MS,MR)}(1,i)\), with source markings in \(V_{(1,i)}^{(MS,MR)}\) ............................................ 110
8.3 The set of arcs, \(A_{(MS,MR)}(2a,i)\), with source markings in \(V_{(2a,i)}^{(MS,MR)}\) ............................................ 110
8.4 The set of arcs, \(A_{(MS,MR)}(2b,i)\), with source markings in \(V_{(2b,i)}^{(MS,MR)}\) ............................................ 111
8.5 The set of arcs, \(A_{(MS,MR)}(3a,i)\), with source markings in \(V_{(3a,i)}^{(MS,MR)}\), for \(MR > 0\). .................. 111
8.6 The set of arcs, \(A_{(MS,MR)}(3b,i)\), with source markings in \(V_{(3b,i)}^{(MS,MR)}\) ............................................ 112
8.7 The set of arcs, \(A_{(MS,MR)}(4,i)\), with source markings in \(V_{(4,i)}^{(MS,MR)}\), for \(MR > 0\). .................. 112
9.1 Determining the Message Channel Bound. ................................................................. 169
9.2 Determining the Acknowledgement Channel Bound. .................................................... 169
9.3 Determining the Bound on the Total Number of Messages and Acknowledgements. .... 170
10.1 \(DFSA_{RG(MS,MR)}\), where rows 4 and 5 are evaluated for \(0 \leq i \leq MS\). .................... 206
10.2 The minimised deterministic FSA, \(MFSA_{RG(MS,MR)}\) representing the protocol language of \(CPN_{(MS,MR)}\). ................................................................. 208
C.1 RG statistics of the CPN in Figs. 5.5 and 5.6. ............................................. 249  
C.2 Successive differences in the number of nodes when \text{MaxSeqNo} = 1. ....................... 254  
C.3 Successive differences in the number of nodes when \text{MaxSeqNo} = 2. ....................... 255  
C.4 Successive differences in the number of nodes when \text{MaxSeqNo} = 3. ....................... 257  
C.5 Successive differences in the number of nodes when \text{MaxSeqNo} = 4. ....................... 258  
C.6 Successive differences in the number of nodes when \text{MaxSeqNo} = 5. ....................... 258  
C.7 Successive differences in the number of arcs when \text{MaxSeqNo} = 1. ....................... 259  
C.8 Successive differences in the number of arcs when \text{MaxSeqNo} = 2. ....................... 261  
C.9 Successive differences in the number of arcs when \text{MaxSeqNo} = 3. ....................... 263  
C.10 Successive differences in the number of arcs when \text{MaxSeqNo} = 4. ...................... 263  
C.11 Successive differences in the number of arcs when \text{MaxSeqNo} = 5. ...................... 264
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABP</td>
<td>Alternating Bit Protocol</td>
</tr>
<tr>
<td>ABRACAD</td>
<td>Alternating Bit sequence numbers, Retransmissions on timeout,</td>
</tr>
<tr>
<td></td>
<td>Acknowledgements, Connection and Disconnection</td>
</tr>
<tr>
<td>ARQ</td>
<td>Automatic Repeat ReQuest</td>
</tr>
<tr>
<td>BRP</td>
<td>Bounded Retransmission Protocol</td>
</tr>
<tr>
<td>CES</td>
<td>Capability Exchange Signalling</td>
</tr>
<tr>
<td>CFFD</td>
<td>Chaos-Free Failures Divergences</td>
</tr>
<tr>
<td>CPN</td>
<td>Coloured Petri Net</td>
</tr>
<tr>
<td>CPN ML</td>
<td>Coloured Petri Net ML</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CSEC</td>
<td>Computer Systems Engineering Centre</td>
</tr>
<tr>
<td>FAST</td>
<td>Fast Acceleration of Symbolic Transition Systems</td>
</tr>
<tr>
<td>FIFO</td>
<td>First-In-First-Out</td>
</tr>
<tr>
<td>FSA</td>
<td>Finite State Automata/Automaton</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>GDN</td>
<td>Global Declaration Node</td>
</tr>
<tr>
<td>HLPN</td>
<td>High-level Petri Net</td>
</tr>
<tr>
<td>HLPNG</td>
<td>High-level Petri Net Graph</td>
</tr>
<tr>
<td>IOTP</td>
<td>Internet Open Trading Protocol</td>
</tr>
<tr>
<td>ITU-T</td>
<td>Telecommunications Standardization Sector of the International</td>
</tr>
<tr>
<td></td>
<td>Telecommunication Union</td>
</tr>
<tr>
<td>LOTOS</td>
<td>Language Of Temporal Ordering Specifications</td>
</tr>
<tr>
<td>MaxRetrans, MR</td>
<td>Maximum number of Retransmissions</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSeqNo, MS</td>
<td>Maximum Sequence number</td>
</tr>
<tr>
<td>ML</td>
<td>Meta-Language</td>
</tr>
<tr>
<td>OG</td>
<td>Occurrence Graph</td>
</tr>
<tr>
<td>RG</td>
<td>Reachability Graph</td>
</tr>
<tr>
<td>SCC</td>
<td>Strongly Connected Component</td>
</tr>
<tr>
<td>SDL</td>
<td>Specification and Description Language</td>
</tr>
<tr>
<td>SML</td>
<td>Standard ML</td>
</tr>
<tr>
<td>SML/NJ</td>
<td>Standard ML of New Jersey</td>
</tr>
<tr>
<td>SWP</td>
<td>Stop-and-Wait Protocol</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TReX</td>
<td>Tool for Reachability Analysis of Complex Systems</td>
</tr>
<tr>
<td>TRL</td>
<td>Timed Rewriting Logic</td>
</tr>
<tr>
<td>UniSA</td>
<td>University of South Australia</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
</tbody>
</table>
Summary

This thesis investigates a method for tackling the verification of parametric systems, systems whose behaviour may depend on the value of one or more parameters. The range of allowable values for such parameters may, in general, be large or unknown. This results in a large number of instances of a system that require verification, one instance for each allowable combination of parameter values. When one or more parameters are unbounded, the family of systems that require verification becomes infinite.

Computer protocols are one example of such parametric systems. They may have parameters such as the maximum sequence number or the maximum number of retransmissions. Traditional protocol verification approaches usually only analyse and verify properties of a parametric system for a small range of parameter values. It is impossible to verify in this way every concrete instance of an infinite family of systems. Also, the number of reachable states tends to increase dramatically with increasing parameter values, and thus the well-known state explosion phenomenon also limits the range of parameters for which the system can be analysed.

In this thesis, we concentrate on the parametric verification of the Stop-and-Wait Protocol (SWP), an elementary flow control protocol. We have used Coloured Petri Nets (CPNs) to model the SWP, operating over an in-order but lossy medium, with two unbounded parameters: the maximum sequence number; and the maximum number of retransmissions.

A novel method has been used for symbolically representing the parametric reachability graph of our parametric SWP CPN model. This parametric reachability graph captures exactly the infinite family of reachability graphs resulting from the infinite family of SWP CPNs. The parametric reachability graph is represented symbolically as a set of closed-form algebraic expressions for the nodes and arcs of the reachability graph, expressed in terms of the two parameters.

By analysing the reachability graphs of the SWP CPN model for small parameter values, structural regularities in the reachability graphs were identified and exploited to develop the appropriate algebraic expressions for the parametric reachability graph. These expressions can be analysed and manipulated directly, thus the properties that are verified from these expressions are verified for all instances of the system.

Several properties of the SWP that are able to be verified directly from the parametric reachability
graph have been identified. These include a proof of the size of the parametric reachability graph in terms of both parameters, absence of deadlocks (undesired terminal states), absence of livelocks (undesirable cycles of behaviour from which the protocol cannot escape), absence of dead transitions (actions that can never occur) and the upper bounds on the content of the underlying communication channel. These are verified from the algebraic expressions and thus hold for all parameter values.

Significantly, language analysis is also carried out on the parametric SWP. The parametric reachability graph is translated into a parametric Finite State Automaton (FSA), capturing symbolically the infinite set of protocol languages (i.e. sequences of user observable events) by means of similar algebraic expressions to those of the parametric reachability graph. Standard FSA reduction techniques were applied in a symbolic fashion directly to the parametric FSA, firstly to obtain a deterministic representation of the parametric FSA, then to obtain an equivalent minimised FSA. It was found that the determinisation procedure removed the effect of the maximum number of retransmissions parameter, and the minimisation procedure removed the effect of the maximum sequence number parameter. Conformance of all instances of the SWP over both parameters to its desired service language is proved.

The development of algebraic expressions to represent the infinite class of Stop-and-Wait Protocols, and the verification of properties (including language analysis) directly from these algebraic expressions, has demonstrated the potential of this method for the verification of more general parametric systems. This thesis provides a significant contribution toward the development of a general parametric verification methodology.
Declaration

I declare that:

- this thesis presents work carried out by myself and does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university;

- to the best of my knowledge it does not contain any materials previously published or written by another person except where due reference is made in the text; and

- all substantive contributions by others to the work presented, including jointly authored publications, is clearly acknowledged.

Guy Edward Gallasch
Acknowledgements

The first person that I would like to thank is my principal supervisor, Professor Jonathan Billington. It is with a deep sense of gratitude that I acknowledge his boundless enthusiasm and dedication, and the influence this has had on me throughout my candidature. His enthusiasm, dedication, thoroughness and professionalism have been a great example to me in my development as a researcher. His incredible expertise and background knowledge in the areas of protocol verification and formal modelling has provided me with invaluable guidance in the areas of my study. His copious amounts of feedback on early papers, early ideas and early directions, as well as his stimulating and incisive suggestions, have been invaluable throughout my candidature. I also thank him for the many spirited and enjoyable discussions we have had on all manner of topics, in particular for providing me with an opportunity to learn from his wealth of knowledge and experience, and also for the chance to work on a variety of research projects he has attracted to the Computer Systems Engineering Centre. His efforts have culminated in my ability to complete this thesis.

I would like to thank the two people that have filled an associate supervisory role at various stages throughout my candidature. Firstly, to Associate Professor Lars Kristensen, now at Aarhus University, Denmark, who provided me with a shining example early in my candidature, and for the many joint publications and the highly enjoyable collaborations we have undertaken. Secondly, to Dr. Jörn Freiheit, now at the Max Planck Institute, Germany, for our many lunchtime discussions and his advice and assistance in numerous joint research projects.

I have been fortunate to be around other high quality and hard working research students in the Computer Systems Engineering Centre during my candidature, with whom I have discussed ideas and collaborated with on other projects in the area of formal verification. They have provided a welcome distraction for me and allowed me to exercise my mind on other interesting problems, often providing insights into my own research. To Dr. Lin Liu, Dr. Nimrod Lilith, Dr. Chun Ouyang, Somsak Vanit-Anunchai, and Cong Yuan, many thanks for the many challenges and great company and support you have provided. A special thankyou is in order for Dr. Lin Liu, for suggesting that I investigate the epsilon-closure approach for removal of epsilon moves from my parametric FSAs.

The work for this thesis could not have been realised without financial support from the following.
ACKNOWLEDGEMENTS

sources: an Australian Postgraduate Award, a top-up scholarship offered by the Computer Systems Engineering Centre, a Division of Information Technology, Engineering and the Environment Small Grant, and School of Electrical and Information Engineering and Computer Systems Engineering Centre financial support for travel and attendance at numerous international conferences. I must also thank Professor Billington for providing me with his assistance in attaining financial support and for providing me with these opportunities.

I am grateful to my family for their support, and especially to my parents, for never applying pressure for me to succeed but for supporting me in whatever endeavour I undertake, and for being there through the highs and the lows along the way.

I would also like to thank my friends, Shaun, Corrie, Mathew, Zoe, Leigh, Sarah, Rhaddie and Jane, for their support, friendship, and for dragging me away from my studies occasionally. Finally, I would like to say a special thank you to Leonie, for her constant support and for being patient and understanding during the final write-up phase of this thesis.
Chapter 1

Introduction

Parametric systems can be considered as any system which has a set of values that are not fixed \textit{a priori} [79] but which may take on different values. Each allowable combination of these values defines a concrete instance of the system. For example, the parameter may be the number of identical instances of an entity, such as the number of philosophers in a parameterised dining philosophers problem [94, 95], the number of processors or bus clients in a cache coherence protocol [107, 119, 131], or the number of entities in a mutual exclusion algorithm [25] or the well-known distributed database problem [81]. These systems share the common theme of having the parameter specify the number of instances of an identical component or entity.

A second class of parametric systems is one in which the parameter values affect the behaviour of the system, but the system structure (i.e. the number of components) remains fixed over all parameter values. Examples of this nature include communication protocols, e.g. the Stop-and-Wait and Sliding Window protocols [127, 135]. The variants of such protocols that are analysed may have as parameters the maximum number of retransmissions, the maximum sequence number (thereby defining a finite sequence number space), maximum sender and receiver window sizes, or an upper bound on the number of messages and acknowledgements in the underlying channels, e.g. [65, 85, 86, 100, 144]. These parametric systems represent an (infinite) class of concrete systems, one for each combination of parameter values.

1.1 Background and Motivation

Stop-and-Wait is an elementary form of flow control [127, 135] used by communication protocols to prevent buffer overflow in the receiver. After sending a message, the sender must stop and wait for an acknowledgement from the receiver before it can send the next message. When the Stop-and-Wait Protocol (SWP) operates over noisy channels, acknowledgements are also used for transmission error recovery. In this case, a checksum is used to detect transmission errors in the message or acknowledgement. If
1.1. Background and Motivation

An error is detected, the message (or acknowledgement) is discarded. A timeout/retransmission scheme, such as Automatic Repeat ReQuest (ARQ) [135], is used to recover from this loss. Sequence numbers are used to detect (and then discard) duplicates of previous messages.

The Stop-and-Wait mechanism forms the basis of many practical data transfer protocols, such as the Internet’s Transmission Control Protocol (TCP) [120]. An understanding of how these mechanisms work and how they may fail is essential for the verification of more complex protocols like TCP. These protocols have a number of parameters, as mentioned above, and thus can be considered parametric systems. The value of these parameters may vary depending on the application (e.g. TCP has a 32 bit sequence number, whereas the X.25 protocol [77] allows the use of 3 bit, 7 bit and 15 bit sequence numbers). It is thus of interest to verify these protocols for all values of these parameters.

Model checking [33] is a well known method for formal verification of systems. It involves generation of all or part of the reachable states of the system (i.e. reachability analysis), in order to prove properties of that system. The key advantage of model checking over other techniques such as theorem proving (e.g. [41,58]) is that the procedures can be relatively easily automated in computer tools such as SPIN [67, 110] or Design/CPN [31, 39] and CPN Tools [38] for Coloured Petri nets [81, 89].

It does, however, have one major disadvantage that has been the focus of much research. This is the well known state explosion problem [143], where even for relatively simple systems, the number of states can be very large. The unfortunate result of state explosion is that in many cases, the entire state space is too large to fit into available computer memory. This means that we cannot use conventional model checking algorithms to reason about many practical systems.

The state explosion phenomenon has motivated the development of a number of techniques to alleviate the problem. A good survey of such techniques can be found in [143]. Broadly, they can be classified into three main categories: techniques that represent the state space in a condensed or compact form, such as symmetry reduction [34,43,80]; techniques that explore only a subset of the reachable states, including partial order methods [114,141,153] such as Valmari’s stubborn sets; and techniques that delete or throw away states or state information during exploration, including bit-state hashing [70, 71, 154], state space caching [59, 68, 69], the pseudo-root technique [113], and more recently the Sweep-line method [32,90–92]. These techniques make it possible to verify larger systems than was previously possible, and can sometimes handle systems with an infinite state space. However, none of the techniques mentioned above explicitly address the issue of parametric systems, i.e. the verification of all instances of the (infinite) families of systems with one concrete instantiation per combination of parameter values.

The goal of this research was to investigate advanced reachability analysis and state space reduction techniques [47]. A substantial period of time was spent investigating the Sweep-line method. However, the results of those investigations have been omitted from this thesis, but have been reported in a number of other papers [47].
of publications [18, 52, 53, 55–57, 147]. Appendix D includes the abstracts of these publications. A Coloured Petri Net model of the Stop-and-Wait Protocol was developed as an example to illustrate the Sweep-line method for [18] although this model did not appear in the final version of [18].

Through experimentation on this model, a number of interesting behavioural properties were discovered [15–17], including the unboundedness of channels and data loss and duplication when operating with retransmissions over a reordering medium, and a bound on the channel content that was dependent on the maximum number of retransmissions and maximum sequence number when operating over an in-order medium. At this point, investigations into the verification of the Stop-and-Wait Protocol became a research topic in itself. Inspired by the work of Gordon [61, 63], Lin [100, 102], and Han [21, 65], we began to look at the parametric verification of the SWP, firstly in one parameter [48, 49] (maximum sequence number, with no retransmissions) and then in both parameters [50, 51].

A large volume of literature (e.g. [4, 5, 9, 22, 40, 122, 129, 133–135, 139]) exists that claims to verify the SWP (or variations) in a non-symbolic way. The Protocol Verification Methodology developed in [17, 23], is adequate for finite systems with tractable reachability graphs, but fails when considering parameterised systems where the range of the parameters is large or unknown. To this end, there is a substantial body of literature (e.g. [10, 25, 79, 94, 95, 107, 123, 131]) concerning methods to solve the parametric analysis problem, based on logics, automata composition and induction. Mostly, these address the first class of parameterised systems, where there are a number of identically behaving components in the system, by taking advantage of the identical behaviour of the components (see e.g. [119, 156] for surveys of techniques). There is surprisingly little to be found in the literature on verification of the second class of parametric system, where the parameter influences behaviour of a fixed system. However, none address the parametric verification of the SWP for all explicit values of the maximum number of retransmissions and maximum sequence number parameters.

1.2 Research Aims

The primary aim of this thesis is to analyse and verify parametrically the infinite class of Stop-and-Wait Protocols operating over an in-order medium. This verification is to be for all allowable values of the maximum sequence number and maximum number of retransmissions parameters. To achieve this, three objectives must be reached.

In order to verify the Stop-and-Wait Protocol for all values of its parameters, a formal specification of the Stop-and-Wait Protocol and of the service it provides to its users (the Stop-and-Wait Service) must be developed. The first objective is thus to provide such specifications.

The second objective is to generate a symbolic representation of the reachable states and state changes
of the parametric system, i.e. a parametric reachability graph. A reachability graph incorporates both the reachable states of the system (i.e. the state space forms the nodes of the graph) and all possible changes of state (i.e. arcs between states). The parametric reachability graph must capture exactly the reachability graph of each member of the infinite class of Stop-and-Wait Protocols.

The third objective is to perform verification of the infinite class of Stop-and-Wait Protocols. A number of properties must be specified and verified parametrically, including that it is a faithful refinement of the Stop-and-Wait Service, so that they are verified for every allowable combination of the maximum sequence number and maximum number of retransmissions parameters.

1.3 Scope

The focus of this thesis is the verification of a parameterised version of the Stop-and-Wait flow control protocol. We only consider the SWP operating over a lossy, in-order underlying medium without duplication, with only one sender and only one receiver. We do not consider connection management activities but rather consider data transfer in isolation, assuming that a connection between the sending and receiving entities has been established previously. Users of the sending and receiving entities are not explicitly modelled and communication between the sending and receiving entities and the users are considered as being synchronised with events in the sender and receiver. We do not consider time, but rather we focus on the logical correctness of the protocol by modelling possible sequences of actions. The parametric verification dealt with in this thesis is of the second class, where the parameter value affects the behaviour of the components, rather than the number of identical components.

The protocol is modelled using Coloured Petri Nets (CPNs) [81, 89]. Both the maximum number of retransmissions (MaxRetrans) and the maximum sequence number (MaxSeqNo) are parameters of our SWP CPN model. CPNs are a suitable modelling language for distributed and concurrent systems such as protocols, primarily because of their ability to express system characteristics such as non-determinism, concurrency, and synchronisation, and the significant computer tool support offered by tools such as Design/CPN [39] and CPN Tools [38].

1.4 Overview of this Thesis

This thesis is organised into the following chapters:

Chapter 2 (Formal Techniques) presents some background information on the formal methods used in this thesis, namely Coloured Petri nets and High-level Petri Net Graphs, and Finite State Automata.
Chapter 3 (Protocol Verification Methodology: An Overview) presents an existing protocol verification methodology, based on formal modelling of both the service specification and the protocol specification. The drawbacks of this methodology in the parametric domain and a method for adaption are discussed.

Chapter 4 (Previous Work) clarifies our notion of parametric in the context of previous work. This chapter provides a survey of previous work, both our own and the work of others, on the verification of the Stop-and-Wait protocol and related protocols using both non-symbolic and symbolic techniques.

Chapter 5 (Stop-and-Wait Flow Control Protocol) introduces the Stop-and-Wait protocol and associated concepts, such as the Stop-and-Wait service. Our parameterised Coloured Petri net model of the Stop-and-Wait protocol is presented, along with a list of modelling assumptions and simplifications and a discussion of the modelling decisions that were made. Our model, formulated as a parameterised High-level Petri Net Graph, is also presented. This chapter also specifies the list of properties of the Stop-and-Wait protocol that we wish to verify.

Chapter 6 (Model Behaviour) investigates the behaviour of our parameterised Stop-and-Wait Protocol model for concrete values of the two parameters, initially when there are no retransmissions and then when retransmissions are allowed. Patterns evident from the reachability graphs of concrete instances of the models, as parameter values increase, are discussed as a basis for the formalisation of the parametric reachability graph. Other behavioural properties of the parameterised Stop-and-Wait Protocol model, including the sequence number independent nature of the model actions, are formalised.

Chapter 7 (A Notation for Markings and Arcs) defines the parametric notation used for representing markings (states) and arcs (transition occurrences) and the notation for the parametric reachability graphs used in the following chapters.

Chapter 8 (The Parametric SWP Reachability Graph) presents the parametric reachability graph, in terms of disjoint subsets of markings and arcs expressed as algebraic expressions in terms of the symbolic parameter values. A proof of the correctness of the parametric reachability graph is given, thus proving that our parametric reachability graph describes exactly the reachability graph of each of the infinite number of concrete instantiations of the CPN model.

Chapter 9 (Analysis of the Parametric Reachability Graph) and Chapter 10 (Parametric Language Analysis) contain the parametric analysis of the Stop-and-Wait protocol model. Chapter 9 begins with a derivation of the size of the parametric reachability graph, in terms of the two parameters,
directly from the algebraic expressions. The properties for verification, defined in Chapter 5, are then verified systematically, except for language analysis, which is carried out in Chapter 10.

Chapter 11 (Conclusions and Future Work) presents a discussion of the results in the context of the aims and contribution of this thesis. A number of future work topics are suggested.

The thesis also includes four appendices. Appendix A introduces concepts from algebraic specification related to the introduction of High-level Petri Net Graphs in Chapter 2. Appendix B presents concrete reachability graphs of the Stop-and-Wait Protocol model obtained from Design/CPN. Appendix C presents Design/CPN output for results regarding the size of the parametric reachability graph. Finally, Appendix D presents a summary of the refereed journal, conference and workshop publications produced during the PhD candidature.
Chapter 2

Formal Techniques

Formal methods [35] 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 [65]).

In this thesis, we make use of two formal techniques, Coloured Petri nets and Finite State Automata. Section 2.1 introduces Coloured Petri nets and High-level Petri Net Graphs, along with associated concepts such as multisets and reachability graphs. Finite State Automata are introduced in Section 2.2, along with related topics including languages and automata reduction techniques.

2.1 Coloured Petri nets

Petri nets [109, 116, 117, 121] are a formal method that have proved adept at modelling systems with concurrency and nondeterminism. Coloured Petri nets (CPNs) [81] are a form of High-level Petri net (HLPN) [73, 84] that provide an executable modelling language with formal semantics, based on Petri nets and the Standard ML (SML) functional programming language [128,140]. High-level Petri nets are defined in international standard ISO/IEC 15909-1 [73] and provides an elegant way of describing the semantics of HLPNs in a generic framework. It is drawn from similar work published in [11].

CPNs and High-level Petri nets in general were developed to model large systems in a much more convenient and compact way. This is particularly true of network protocols, which exhibit a large degree of concurrency and nondeterminism, e.g. see [14]. Of particular relevance to this thesis is their success at modelling and analysing communication and transaction protocols, e.g. [60,65,83,111,148]. Both CPNs and the HLPNG formalism are used in this thesis, thus to make the thesis self-contained, we introduce the formal definition of High-level Petri Net Graphs (HLPNG) from [73]. CPNs are then introduced informally in this context by way of a small example, which we then formalise as a HLPNG. A good
tutorial introduction to CPNs is given by Kristensen et al [89].

The following definitions are based on those presented in [73] (except where indicated). We begin by presenting the definition of a multiset and some operations on multisets, followed by the HLPNG definition and the associated concepts of markings, the enabling rule and the transition (occurrence) rule for a HLPNG. We assume familiarity with the concept of sets.

2.1.1 Multisets

Definition 2.1 (Multisets).
A multiset, $B$, over a non-empty basis set, $A$, is a function

$$B : A \rightarrow \mathbb{N}$$

which associates a multiplicity (a natural number), possibly zero, with each of the basis elements. The multiplicity of $a \in A$ in $B$, is given by $B(a)$. A set is a special case of a multiset, where the multiplicity of each of the basis elements is either zero or one.

The set of all multisets over $A$ is denoted by $\mathcal{A}$. As an example, $\{3, 1\}, (7, 1), (i, 0) \mid i \in \mathbb{N} \setminus \{3, 7\}$ is a multiset over the natural numbers, $\mathbb{N}$. Note that for convenience, we often omit any element of the basis set with a multiplicity of zero from the multiset expression, so that this example multiset could be written as $\{3, 1\}, (7, 1)$.  

Definition 2.2 (Sum Representation).
A multiset may be represented as a symbolic sum of basis elements scaled by their multiplicities (sometimes known as co-efficients).

$$B = \sum_{a \in A} B(a) \cdot a$$

For example, the multiset $\{3, 1\}, (7, 1)$ could be written as $1^3 + 1^7$, where $+$ represents the symbolic addition of basis elements scaled by their multiplicities, and (as before) any element of the basis set with multiplicity of zero is omitted from the sum representation.

Definition 2.3 (Membership).
Given a multiset, $B \in \mu A$, $a \in A$ is a member of $B$, denoted $a \in B$, if $B(a) > 0$, and conversely if $B(a) = 0$, then $a \notin B$.

Definition 2.4 (Empty Multiset).
The empty multiset, $\emptyset$, has no members: $\forall a \in A, \emptyset(a) = 0$.

Definition 2.5 (Cardinality).
2.1. Coloured Petri nets

The cardinality, $|B|$, of a multiset, $B \in \mu A$, is the sum of the multiplicities of each of the members of the multiset.

$$|B| = \sum_{a \in A} B(a)$$

When $|B|$ is finite, the multiset is called a finite multiset.

2.1.2 Operations on Multisets

Definition 2.6 (Equality and Comparison).

Two multisets, $B_1, B_2 \in \mu A$, are equal, $B_1 = B_2$, iff $\forall a \in A, B_1(a) = B_2(a)$.

$B_1$ is less than or equal to (or contained in) $B_2$, $B_1 \leq B_2$, iff $\forall a \in A, B_1(a) \leq B_2(a)$.

$B_1$ is greater than or equal to $B_2$, $B_1 \geq B_2$, iff $\forall a \in A, B_1(a) \geq B_2(a)$.

Definition 2.7 (Addition and Subtraction).

The addition and subtraction operations on multisets, $B_1, B_2 \in \mu A$, associate to the left and are defined as follows:

$$B = B_1 + B_2 \text{ iff } B(a) = B_1(a) + B_2(a)$$

$$B = B_1 - B_2 \text{ iff } B(a) = B_1(a) - B_2(a) \text{ if } B_1(a) \geq B_2(a)$$

Note that subtraction is a partial operation.

Definition 2.8 (Singleton Multiset to Element (based on [65])).

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 $f_c((a, 1)) = a$.

2.1.3 High-level Petri Net Graph

Definition 2.9 (High-level Petri Net Graph).

A High-level Petri Net Graph is the structure $\text{HLPNG} = (NG, \text{Sig}, Vars, Alg, Type, AN, M_0)$ where

- $NG = (P, T; F)$ is called a Net Graph, with
  - $P$ a finite set of nodes, called Places;
  - $T$ a finite set of nodes, called Transitions, disjoint from $P$ (i.e. $P \cap T = \emptyset$); and
  - $F \subseteq (P \times T) \cup (T \times P)$ a set of directed edges called arcs, known as the Flow relation.
- $\text{Sig} = (Sorts, Ops)$ is a Boolean signature defined in Appendix A.
2.1. Coloured Petri nets

- \( Vars \) is a \( \text{Sorts} \)-indexed set of variables, disjoint from \( Ops \).

- \( Alg = (\text{Types}, \text{Funs}) \) is a many-sorted algebra for the signature \( \text{Sig} \), defined in Appendix A.

- \( Type : P \rightarrow \text{Types} \) is a function which assigns carriers (types) to places.

- \( AN = (\text{ATerm}, \text{TCond}) \) is a pair of net annotations, with

  - \( \text{ATerm} : F \rightarrow \text{TERM}(\text{Ops} \cup \text{Vars}) \) such that for all \( (p, t), (t', p) \in F \), for all bindings \( \alpha, \text{Val}_\alpha(\text{ATerm}(p, t)), \text{Val}_\alpha(\text{ATerm}(t', p)) \in \mu\text{Type}(p) \). \( \text{TERM}(\text{Ops} \cup \text{Vars}) \), \( \alpha \) and \( \text{Val}_\alpha \) are defined in Appendix A. \( \text{ATerm} \) is a function that annotates each arc with a term that when evaluated (for any binding) results in a multiset over the associated place’s type.

  - \( \text{TCond} : T \rightarrow \text{TERM}(\text{Ops} \cup \text{Vars})_{\text{Bool}} \) is a function that annotates transitions with Boolean expressions (Transition Conditions).

- \( M_0 : P \rightarrow \bigcup_{p \in P} \mu\text{Type}(p) \) such that \( \forall p \in P, M_0(p) \in \mu\text{Type}(p) \), is the initial marking function. It associates a multiset of tokens (of the correct type) with each place.

Marking of a HLPN

**Definition 2.10** (Marking of a HLPNG).

A marking, \( M \), of the HLPNG is defined in the same way as the initial marking. \( M : P \rightarrow \bigcup_{p \in P} \mu\text{Type}(p) \) such that for all \( p \in P, M(p) \in \mu\text{Type}(p) \).

Enabling of Transition Modes

**Definition 2.11** (Enabling of Transition Modes).

A transition \( t \in T \) is enabled in a marking, \( M \), for a particular assignment, \( \alpha_t \), to its variables, that satisfies the transition condition, \( \text{Val}_{\text{Bool},\alpha_t}(\text{TCond}(t)) = \text{true} \), known as a mode of \( t \), iff

\[
\forall p \in P, \text{Val}_{\alpha_t}(\overrightarrow{p, t}) \leq M(p)
\]

where for \( (u, v) \in (P \times T) \cup (T \times P) \),

- \( \overrightarrow{u, v} = \text{ATerm}(u, v), \) for \( (u, v) \in F \),

- \( \overrightarrow{u, v} = \Phi, \) for \( (u, v) \notin F \)

where \( \Phi \) is a symbol that represents the empty multiset at the level of the signature, so that \( \text{Val}_{\alpha_t}(\Phi) = \emptyset \).

We can also define the concurrent enabling of a finite multiset of transition modes (see [73]) but this is not required for this thesis.
2.1. Coloured Petri nets

![Figure 2.1: A simple Coloured Petri Net example.](image)

```
1 color INT = int; (* Integer colour set *)
2 color BOOL = bool; (* Boolean colour set *)
3 var x, y : INT; (* Integer variables *)
```

Transition Rule

**Definition 2.12** (Transition Rule).

*If* \( t \in T \) *is enabled in mode* \( \alpha_t \), *for marking* \( M \), *\( t \) may occur in mode* \( \alpha_t \). *When* \( t \) *occurs in mode* \( \alpha_t \), *the marking of the net is transformed to a new marking* \( M' \), *denoted* \( M[t, \alpha_t]M' \), *according to the following rule:

\[
\forall p \in P, M'(p) = M(p) - Val_{\alpha_t}(\overline{p}, t) + Val_{\alpha_t}(\overline{t}, p)
\]

We can also define the concurrent occurrence of a finite multiset of transition modes (see [73]) but this is not required for this thesis.

### 2.1.4 A Simple Coloured Petri Net Example

To informally introduce CPNs, we present a simple example. Consider a machine, or process, that takes two integers as its input. It performs two operations on these integers: addition; and a comparison to determine if the first is less than the second. Both results are recorded. The CPN in Figs. 2.1 and 2.2 could represent such a machine or process. The software tool, Design/CPN [39], was used for the construction of the model. Later, we will formalise this CPN as a HLPNG.
2.1. Coloured Petri nets

Figure 2.1 presents the graphical representation of the CPN, while Fig. 2.2 defines all the sets required and declares the types of the variables used in the inscriptions associated with the graphical representation. It is also possible to define constants and functions, although this is not necessary for this example. (The functions for integer addition, $+$, and integer “less than” and “not equals”, $<$ and $<=$, and the integer constant, 3, are all pre-defined by Design/CPN.) CPN ML [37], a variant of the functional programming language Standard ML of New Jersey (SML/NJ) [128], is used for the net inscriptions in Fig. 2.1 and the declarations in Fig. 2.2.

The graphical elements of a CPN diagram comprise a bipartite directed graph, consisting of two types of nodes, places and transitions. These correspond to $P$ and $T$ respectively, of the net graph, $NG$, in Definition 2.9. By convention [81], places are represented by ellipses and transitions are represented rectangles. The convention used in this thesis is to position the names given to places and transitions inside the corresponding place or transition. Directed arcs link places to transitions (incoming or input arcs) and transitions to places (outgoing or output arcs). This corresponds to the flow relation, $F$, of the net graph, $NG$, in Definition 2.9. Places connected to transitions via incoming arcs are called input places. Places connected to transitions via outgoing arcs are called output places.

Places may contain tokens. In CPNs tokens are arbitrarily complex data values. Accordingly, a place in a CPN is typed by an appropriate set of data values, called a colour set. The colour set defines all possible data values, or colours, that tokens residing on the corresponding place are able to take. The set of colour sets corresponds to the set of types, $Types$, in Definition 2.9. Every place must be typed by a non-empty colour set, usually written in italics below the place, to the left or right as room permits. The typing of places corresponds to the typing function, $Type$, in Definition 2.9. To illustrate, Fig. 2.2 defines two colour sets, $INT$ and $BOOL$, representing the sets of integers and booleans respectively. In Fig. 2.1, places $Input1$, $Input2$ and $Sum$ are typed by the colour set $INT$, and the $Comparison$ place is typed by the colour set $BOOL$.

The marking of a place in a CPN is given by the function $M(place name)$ which takes a place name and returns the multiset of tokens on that place. The vector of markings of all places of the CPN is called the marking of the CPN, denoted $M$. This corresponds to the marking, $M$, given in Definition 2.10, which maps from places to the set of multisets over the type of the corresponding place. Each place may be given an initial marking, corresponding to $M_0$ in Definition 2.9. By convention, the initial marking is written above the place and usually on the same side as the colour set inscription. The $Input1$ place in Fig. 2.1 has an initial marking of $1^{3} + 1^{7}$. This is read as “a multiset comprising one token of colour 3 and one token of colour 7”. For cases such as this, where the multiplicity of a token colour is 1, it is possible to omit the 1$^{\dagger}$ from the inscription.

As in HLPNGs, arcs may be annotated with arbitrarily complex expressions involving constants,
variables and functions (operators), provided that each arc expression evaluates to a multiset over the
colour set of the corresponding (input or output) place, for every valid binding (assignment) of values to
these variables. This corresponds to the arc annotation function, $ATerm$, of the net annotations, $AN$,
in Definition 2.9. For example, we can see that the arc from transition $AddAndCompare$ to place $Sum$
has an expression containing variables $x$ and $y$ and the integer addition function. This expression will
always evaluate to a single token from the colour set $INT$, for any valid binding of values to the variables
$x$ and $y$ (see Definition 2.11 and Appendix A.6). Again we note that when the multiplicity of this token
is one, the 1' is omitted from the graphical representation.

Each transition in a CPN has a boolean expression called a guard associated with it. The guard may
use constants, operators, any of the variables present in the arc expressions of the incoming arcs of the
transition in question, and may also introduce new variables that are local to the guard. The guard is
usually written near the transition and by convention is enclosed in square brackets. This corresponds
to the transition condition function, $TCond$, of the net annotations, $AN$, in Definition 2.9. For example,
the transition $AddAndCompare$ has the guard $[x <> 3]$ with the variable, $x$, the constant, 3, and the
“not equals to” operator, $<>$. The default guard expression ($[true]$) is usually omitted from the graphical
representation.

The execution of a CPN consists of a sequence of occurrences (or firings) of transitions. A transition
can occur if it is enabled. For a particular marking of the CPN, a transition is enabled in a mode,
determined by the binding of its variables, when the following conditions are true:

1. Each input place of the transition contains at least the tokens obtained when its input arc expression
   is evaluated for the specific binding; and

2. The guard of the transition evaluates to true for that binding.

This is analogous to the enabling of transition modes of HLPNGs given in Definition 2.11. In CPN termi-
nology, the combination of a transition and a specific binding of its variables, i.e. a transition enabled in
a mode, is called a binding element [81]. A binding element can be represented as a pair, $(t, b)$, where $t$ is
the name of a transition and $b$ is a binding of its variables. For example, $(AddAndCompare, (x = 7, y = 2))$
is a binding element of the CPN in Fig. 2.1, enabled in the initial marking. In CPN notation [81],
binding elements are written as $t<b>$, e.g. $AddAndCompare<x = 7, y = 2>$.

When a transition occurs, the following two actions occur atomically:

1. The multiset of tokens corresponding to the evaluation of each input arc expression is removed
   from the corresponding input place; and

2. The multiset of tokens corresponding to the evaluation of each output arc expression is added to
   the corresponding output place.
2.1. Coloured Petri nets

This is analogous to the transition rule of HLPNGs given in Definition 2.12. If the binding element, \texttt{AddAndCompare} < \(x = 7, y = 2\) >, were to occur in the initial marking, the resulting marking would be given by \(M(\text{Input1}) = 1'3, M(\text{Input2}) = 1'9, M(\text{Sum}) = 1'9\) and \(M(\text{Compare}) = 1'\text{true}\).

Transition modes (binding elements) may be \textit{concurrently enabled} with one another, and also with themselves, by a linear extension of the above enabling conditions and occurrence rules [81]. However, as with the HLPNG definition, this is not required for this thesis.

A marking in which there are no transitions enabled is called a \textit{dead marking}, or a \textit{terminal marking} of the system. This is a marking from which no further actions can occur. Dead markings that are undesirable are sometimes called \textit{deadlocks}.

Every CPN model contains a set of declarations, comprising declarations for variables, and definitions for colour sets, functions and constants. In Design/CPN, this is called a \textit{Global Declaration Node} (GDN). The GDN for this CPN model is shown in Fig. 2.2. Colour sets (types) are defined using the keyword \texttt{color} and a number of set constructors like \texttt{with} (for enumerated types), \texttt{product} and \texttt{union}. Variables are declared by the keyword \texttt{var} and are typed by a colour set, e.g. the variables \(x\) and \(y\) of type \texttt{INT}. Functions are defined using the keyword \texttt{fun} and values (e.g. constants) are defined using the keyword \texttt{val}. This is a particularly valuable construct for the development of parameterised CPN models.

2.1.5 The Corresponding High-level Petri Net Graph

In this section we formalise the CPN in Figs. 2.1 and 2.2 as a HLPNG:

- \(NG = (P, T, F)\) where:
  - \(P = \{\text{Input1}, \text{Input2}, \text{Sum}, \text{Comparison}\}\);
  - \(T = \{\text{AddAndCompare}\};\) and
  - \(F = \{(\text{Input1}, \text{AddAndCompare}), (\text{Input2}, \text{AddAndCompare}),
    (\text{AddAndCompare}, \text{Sum}), (\text{AddAndCompare}, \text{Comparison})\}\).

- \(\text{Sig} = (\text{Sorts}, \text{Ops})\) where:
  - \(\text{Sorts} = \{\text{Bool}, \text{Int}\};\) and
  - \(\text{Ops} = \{+(\text{Int}, \text{Int}), <(\text{Int}, \text{Int}, \text{Bool}), >(\text{Int}, \text{Int}, \text{Bool}), 3\text{Int}, \text{trueBool}\}\).

- \(\text{Vars} = \{\text{xInt}, \text{yInt}\}\)

- \(\text{Alg} = (\text{Types}, \text{Funs})\) where:
  - \(\text{Types} = \{\text{BOOL}, \text{INT}\};\) and
2.1. Coloured Petri nets

- \( Funs = \{+, <, <>, 3, true\} \), where
  - \(+ : INT \times INT \to INT\) is the usual integer addition function;
  - \(< : INT \times INT \to BOOL\) is the usual “less than” relationship on integers;
  - \(<> : INT \times INT \to BOOL\) is the usual “not equals to” relationship on integers;
  - \(3 : INT\) is the integer value “three”; and
  - \(true : BOOL\) is the boolean value “true”.

- \(Type(p) = \begin{cases} 
  BOOL, & \text{if } p = \text{Comparison} \\
  INT, & \text{otherwise}
\end{cases}\)

- \(AN = (ATerm, TCond)\) where:

- \(ATerm(u, v) = \begin{cases} 
  x_{Int}, & \text{if } (u, v) = (\text{Input1}, \text{AddAndCompare}) \\
  y_{Int}, & \text{if } (u, v) = (\text{Input2}, \text{AddAndCompare}) \\
  +_{(Int, Int, Int)} (x_{Int}, y_{Int}), & \text{if } (u, v) = (\text{AddAndCompare, Sum}) \\
  <_{(Int, Int, Bool)} (x_{Int}, y_{Int}), & \text{if } (u, v) = (\text{AddAndCompare, Comparison})
\end{cases}\)

- \(TCond(t) = \begin{cases} 
  <_{(Int, Int, Bool)} (x_{Int}, 3_{Int}), & \text{if } t = \text{AddAndCompare}, \\
  true_{Bool}, & \text{otherwise}
\end{cases}\)

- \(\{\{3, 1\}, \{7, 1\}\}, \text{ or } 1'3 + +1'7, \text{ if } p = \text{Input1}\)
- \(\{\{2, 1\}, \{9, 1\}\}, \text{ or } 1'2 + +1'9, \text{ if } p = \text{Input2}\)
- \(\emptyset, \text{ otherwise}\)

Note that in CPNs, the symbols used in expressions on arcs and in guards are terms built from operators and variables. Hence, in Fig. 2.1, the non-constant symbols (+, <> and <) are actually operators and not functions. However, this distinction is very subtle. In this thesis, for simplicity, we shall use the same symbol for each operator and its corresponding function. This also applies to the constants, 3 and true.

2.1.6 Reachability Graphs

Given a CPN with an initial marking, it is possible to construct a graph which captures all states and state changes of the CPN, where each node of the graph represents a marking of the CPN (the initial node of the graph represents the initial marking of the CPN) and each edge in the graph represents the occurrence of an enabled binding element (transition mode). These binding elements become the edge labels. The nodes can be considered as the states of the system and the arcs represent the possible changes in state.
2.1. Coloured Petri nets

Leaf nodes correspond to the dead markings, as they have no outgoing arcs (i.e. no actions are enabled in those states). This graph is called the Reachability Graph (RG), and is also referred to as the Occurrence Graph (OG) [81] because it represents all possible occurrences of transitions. Throughout this thesis we shall use the term Reachability Graph and the corresponding abbreviation, RG.

In this thesis, the notation \([M]\) is used to represent all markings reachable from marking \(M\). The notation \(M \left[ (t, b) \right]\) is used to indicate that the binding element, \((t, b)\), is enabled by marking \(M\). For example, \(M_0 \left[ (\text{AddAndCompare}, (x = 7, y = 2)) \right]\). The notation \(M \left[ t \right]\) is used as shorthand to represent that there exists some binding, \(b\), of the variables of \(t\) such that \(M \left[ (t, b) \right]\). The Reachability Graph of a CPN with initial marking \(M_0\) can now be formalised, after first formally defining the concept of reachability:

**Definition 2.13 (Reachability (based on [65])).**

Let \(M, M_1\) and \(M_2\) be markings of a CPN and \(BE\) be the set of binding elements of the same CPN. We define \([M]\) inductively as follows:

1. \(M \in [M]\); and

2. for \(M_1 \in [M]\), if \(M_1 \left[ (t, b) \right] M_2\) then \(M_2 \in [M]\), where \(M_1 \left[ (t, b) \right] M_2\) denotes that the marking of the CPN changes from \(M_1\) to \(M_2\) on the occurrence of binding element \((t, b)\) \(\in BE\).

Marking \(M'\) is reachable from \(M\), iff \(M' \in [M]\).

**Definition 2.14 (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. \(V = \left[ M_0 \right]\) is the set of reachable markings of the CPN; and

2. \(A = \{(M, (t, b), M') \in V \times BE \times V \mid M \left[ (t, b) \right] M'\}\) is a set of labelled directed arcs.

Reachability graphs can be used to determine many properties of a system, both standard dynamic properties [81] such as absence of deadlock and livelock, and more system-specific properties such as whether two processes can ever be in their critical sections simultaneously. The largest drawback of this technique is that of state explosion [143], the phenomenon of a combinatorial increase in the number of states of the system, such that even moderately sized systems have too many states to fit in computer memory. This provides one of the motivations for the work presented in this thesis. We return to the notion of formalising desired properties of our Stop-and-Wait protocol in Chapter 5 and verify these properties parametrically in Chapters 9 and 10.
2.2 Finite State Automata

2.1.7 Coloured Petri Net Tool Support

CPNs have a high degree of computer support for both their construction and analysis. The computer tool Design/CPN [39], formerly maintained by the University of Aarhus, Denmark, is the computer tool used in conjunction with this thesis. Design/CPN is based on the unix/linux platform.

Design/CPN has four main facilities: an editor, a simulator, a state space tool and a performance tool. The editor supports construction, editing and syntax checking of CPN models. The simulator allows interactive and automatic simulation of CPN models. The state space tool [31] supports reachability analysis of CPN models, including the ability to visualise and manually lay out the nodes and arcs of the reachability graph. This feature is especially useful as will be seen in Chapter 6. Finally, the performance tool [151] is used to support simulation-based performance analysis. The simulation engine and state space tool are built using CPN ML.

Design/CPN has recently been superseeded by a tool called CPN Tools [38], also developed and maintained by the University of Aarhus, based on the Windows platform. There are two reasons why Design/CPN was used in conjunction with this thesis and not CPN Tools. The first is that this work was started using Design/CPN at a time before CPN Tools superseeded Design/CPN. The second is that at the time of writing, CPN Tools lacked the reachability graph visualisation facilities of Design/CPN, which were an integral part of the advancement of the work in this thesis.

In some sense the tool used is not critical, as the tool itself plays only a small part in the work presented in this thesis. The major contribution is in parametric specification and analysis, which is quite independent of any tool.

2.2 Finite State Automata

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, 72].

2.2.1 Languages

The relationship between languages and automata is a close one. We begin by defining the following language concepts and notation, based on [8, 72, 97].

**Definition 2.15** (Alphabet).

*An alphabet is a finite, non-empty set of symbols, conventionally denoted \( \Sigma \).*
2.2. Finite State Automata

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

**Definition 2.16 (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, $\Sigma$, is denoted $\Sigma^+$. 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 $\epsilon$, and may be chosen from any alphabet. The set of all strings, including the empty string, over $\Sigma$ is denoted $\Sigma^*$, 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 2.17 (Language).**

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

For example, $L = \{abcd, efgh\}$ 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 [87]).

**Definition 2.18 (Language Inclusion).**

Given two languages, $L_1$ and $L_2$, defined over the same alphabet, we say that $L_1$ is included in $L_2$ if and only if all strings in $L_1$ are also in $L_2$, i.e. $L_1 \subseteq L_2$.

**Definition 2.19 (Language Equivalence).**

Two languages, $L_1$ and $L_2$, defined over the same alphabet, are equivalent if and only if all strings in $L_1$ are also in $L_2$ and all strings in $L_2$ are also in $L_1$, i.e. $L_1 = L_2$ iff $L_1 \subseteq L_2$ and $L_2 \subseteq L_1$.

2.2.2 Formal Definitions of Finite State Automata

The following formally defines a Finite State Automaton as used in this thesis, based on [13, 87, 97].

**Definition 2.20 (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;
- $\Sigma$ 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 $\epsilon$;
2.2. Finite State Automata

Figure 2.3: A Simple Nondeterministic FSA.

- \( s_0 \in S \) 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 thesis. It covers both nondeterministic and deterministic FSAs, with or without \( \epsilon \) (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. 2.3. 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. 2.3 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 \( \epsilon \), 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 \( \epsilon \) can be traversed without consuming an input symbol.) In Fig. 2.3, note the nondeterministic choice introduced by the \( \epsilon \) 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, 100]:

**Definition 2.2.1 (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 \( s_1, s_2, ..., s_{n-1} \in S \) such that for all \( i \in \{0, ..., n - 1\} \), \((s_i, \ell_i, s_{i+1}) \in \Delta\).
2.2. Finite State Automata

A Deterministic FSA (DFSA) is a special case of Definition 2.20 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 [100]):

**Definition 2.22** (Deterministic Finite State Automaton).
A FSA, \( FSA = (S, \Sigma, \Delta, s_0, F) \), is a DFSA, if its transition relation, \( \Delta \), 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 [72, 87, 97, 99, 126], \( \delta : S \times \Sigma \rightarrow S \), 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.

2.2.3 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, \( \sigma \), 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. 2.3 contains
two strings, ABCDE and ABCYZ. The formal definition of the language of a nondeterministic FSA is not needed in this thesis, but can be found in [72]. The language of a deterministic FSA can be defined formally by firstly defining an extended transition function, inspired by [72, 100].

**Definition 2.23** (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
\[
\hat{\delta}(s, l\sigma) = \begin{cases} 
\delta(s, l), & \text{if } \sigma = \epsilon \\
\hat{\delta}(s, l), & \text{if } \sigma \in \Sigma^+
\end{cases}
\]
where \( s \in S \) and \( l \in \Sigma \), and juxtaposition is used for string concatenation.

Unlike [72], 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 \( \epsilon \) for this purpose may be confusing (recall that we are dealing with a DFSA, with no \( \epsilon \) moves).

**Definition 2.24** (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
\[
\mathcal{L}(DFSA) = \begin{cases} 
\{ \sigma \in \Sigma^+ | \hat{\delta}(s_0, \sigma) \in F \}, & \text{if } s_0 \notin F \\
\{ \sigma \in \Sigma^+ | \hat{\delta}(s_0, \sigma) \in F \} \cup \{ \epsilon \}, & \text{if } s_0 \in F
\end{cases}
\]
2.2. Finite State Automata

Unlike [100], we include the possibility of a DFSA accepting the empty string when the initial state is also a halt state. (In [72] this is implicit in the extended transition function.)

2.2.4 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].

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 [72, 126]. $\epsilon$ closure involves computing the transitive closure of the (not necessarily connected) subgraph of the FSA consisting of all $\epsilon$ 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 [46], which may result in the inclusion of inaccessible states (states not reachable from the initial state) which require deletion. For more details, see [108, 146].

The $\epsilon$-closure operation defines all states in an FSA reachable from a given state via a sequence of 0 or more $\epsilon$ moves. We define the $\epsilon$-closure operation based on the narrative description in [72].

Definition 2.25 ($\epsilon$-closure).

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

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

Definition 2.26 (Linear extension of $\epsilon$-closure).

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

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

Definition 2.27 (non-$\epsilon$ 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-$\epsilon$ outgoing edges of the states in $S'$, is given by $\text{OutEdges} : 2^S \rightarrow 2^\Delta$ where $\text{OutEdges}(S') = \{(s', l, s'') | s' \in S', (s', l, s'') \in \Delta, l \neq \epsilon\}$

If $S'$ is an $\epsilon$-closure (as will be our intention) then this function returns all non-$\epsilon$ outgoing edges of the $\epsilon$-closure.
2.2. Finite State Automata

Determinisation using Subset Construction

A standard subset construction technique is presented in [8, 72, 126] for determinisation of an \( \epsilon \)-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 [72, 126] by combining subset construction with \( \epsilon \)-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 \( \epsilon \)-closure as in [72, 126], given a non-deterministic FSA, \( NDFSA = (S, \Sigma, \Delta, s_0, F) \), an equivalent deterministic FSA, \( DFSA = (S^{\text{det}}, \Sigma, \Delta^{\text{det}}, s_0^{\text{det}}, F^{\text{det}}) \), 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. \( S^{\text{det}} = 2^S \setminus \emptyset \). This may be a large set. Often, in practice, many of the states in \( S^{\text{det}} \) will not be accessible from the start state of DFSA, and so can be effectively thrown away [72], i.e. \( S^{\text{det}} \subseteq 2^S \setminus \emptyset \).

2. Consider state \( s^{\text{det}} = \{s_1, s_2, \ldots, s_n\} \), \( s^{\text{det}} \in S^{\text{det}} \) and \( s_1, s_2, \ldots, s_n \in S \). For each \( l \in \Sigma \), let \( s^{\text{det}}_l = \{s'' \mid (s, l, s') \in \Delta, s'' \in \text{ Closure}(s') \} \), \( s \in s^{\text{det}} \). If \( s^{\text{det}}_l \neq \emptyset \), add to \( \Delta^{\text{det}} \) the arc \( (s^{\text{det}}, l, s^{\text{det}}_l) \). Repeat for all \( s^{\text{det}} \in S^{\text{det}} \).

3. The initial state of \( DFSA \) is the \( \epsilon \)-closure of the initial state of \( NDFSA \), i.e. \( s^{\text{det}}_0 = \text{ Closure}(s_0) \).

4. The final states of \( DFSA \) are all states in \( S^{\text{det}} \) which contain a final state of \( NDFSA \), i.e. for all \( s^{\text{det}} \in S^{\text{det}} \), \( s^{\text{det}} \in F^{\text{det}} \) if and only if \( \exists s \in s^{\text{det}} \) such that \( s \in F \).

Formally:

**Definition 2.28 (Determinisation).**

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

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

The two differences between the subset construction technique combining \( \epsilon \)-closure and the original subset construction technique are [72]:
2.2. Finite State Automata

![Diagram of Finite State Automata](image)

Figure 2.4: The language equivalent Deterministic FSA of the simple Nondeterministic FSA in Fig. 2.3.

1. The initial state of $DFSA$ is the $\epsilon$-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 $\epsilon$-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 $l$.

**Subset Construction using Lazy Evaluation**

A lazy approach to subset construction is also presented in [72]. 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 [72], an algorithm for producing $DFSA$ from $NDFSA$ using subset construction with lazy evaluation can be defined as follows.

1. The initial marking of $DFSA$, $s^{det}_0 = \text{Closure}(s_0)$, is guaranteed to be accessible, i.e. $s^{det}_0 \in S^{det}$.

2. Select a state, $s^{det} = \{s_1, s_2, ..., s_n\}$, 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, $s^{det'} = \text{CLOSEURE}(%\{s' \mid (s, l, s') \in \Delta%\})$. 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 2.4 shows the language equivalent deterministic FSA resulting from applying this algorithm to the FSA in Fig. 2.3. This lazy evaluation method is used symbolically for determinisation in Chapter 10.
2.2. Finite State Automata

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) [72]. The process of obtaining a minimal representation is known as minimisation.

Both [72] 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 [72, 100]:

**Definition 2.29 (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 $Strings_{DFSA}(s) = Strings_{DFSA}(s')$, where $Strings_{DFSA}: S \rightarrow \Sigma^*$ is a function that returns the set of strings accepted by a state of DFSA, defined as:

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

$s$ 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 [72, 100]).

**Definition 2.30 (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 [72] 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 [72], 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
2.2. Finite State Automata

$q$ 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. 2.5. In Fig. 2.5 (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. 2.5 (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 2.31** (Minimised FSA of a Deterministic FSA).

Let $DFSA = (S_{det}, \Sigma, \Delta_{det}, s_{0det}, F_{det})$ be a deterministic FSA. A language equivalent minimum FSA is given by $MFSA = (S_{min}, \Sigma, \Delta_{min}, s_{0min}, F_{min})$, where:

- $S_{min} \subseteq 2^{S_{det}} \setminus \emptyset$ 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 $\forall s_{1det}, s_{2det} \in s_{min}$:
  1. $s_{1det} \in F_{det} \iff s_{2det} \in F_{det}$; and
  2. $\forall l \in \Sigma, (s_{1det}, l, s_{1det}') \in \Delta_{det} \iff (s_{2det}, l, s_{2det}') \in \Delta_{det}$ and $\exists s_{min'} \in S_{min}$ such that $s_{1det}'$ and $s_{2det}' \in s_{min'}$. 


2.3 Concluding Remarks

Figure 2.6: The language equivalent Minimised FSA of the Deterministic FSA in Fig. 2.4.

\[ \Delta^{\text{min}} = \{ (s^{\text{min}}, l, s^{\text{min}}') \mid s^{\text{min}}, s^{\text{min}}' \in S^{\text{min}} \text{ and } (s^{\text{det}}, l, s^{\text{det}}') \in \Delta^{\text{det}} \text{ for some } s^{\text{det}} \in S^{\text{det}}, s^{\text{det}}' \in s^{\text{min}}' \}; \]

- \( s^{\text{min}}_0 \in S^{\text{min}} \mid s^{\text{det}}_0 \in s^{\text{min}}_0 \), and

- \( F^{\text{min}} \subseteq S^{\text{min}} \mid s^{\text{min}} \in F^{\text{min}} \implies s^{\text{min}} \subseteq F^{\text{det}} \)

Figure 2.6 shows the language equivalent minimised FSA resulting from applying the minimisation algorithm to the deterministic FSA in Fig. 2.4.

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 [46]. However, tools from the FSM Library were only used in support of the work in this thesis and not used directly.

2.3 Concluding Remarks

In this chapter, we have introduced Coloured Petri Nets, High-level Petri Net Graphs, and Finite State Automata. Next, we shall see how these three formal techniques can be applied to the verification of protocols, such as the Stop-and-Wait Protocol, as part of a Protocol Verification Methodology.
Chapter 3

Protocol Verification Methodology: An Overview

Protocol engineering [12, 70, 98, 103, 104, 118, 125] is a field that involves the design, implementation, analysis and testing of protocols. This thesis concentrates on one aspect of protocol engineering, namely those activities known as protocol verification. This range of activities has been formalised in a methodology known as the Protocol Verification Methodology [17, 23]. In the context of Coloured Petri nets, this methodology has been successfully applied to many industrial sized examples, e.g. [60, 62, 64–66, 101, 111, 112, 137, 148, 149].

The intention of this chapter is to provide a brief overview of the protocol verification methodology in the context of layered protocol architectures [75], with special focus on those parts that are relevant to the work in this thesis. Section 3.1 provides this overview. Some limitations of the protocol verification methodology with respect to parametric systems are identified and briefly discussed in Section 3.2. For more detailed information on the concepts and procedures involved in protocol verification, the interested reader is referred to [17, 23, 65, 100, 132, 150].

3.1 Overview of the Protocol Verification Methodology

We describe the main steps of the protocol verification methodology from [17] in the following sections.

3.1.1 Service specification

In a layered protocol architecture the service specification is a formal document which specifies the service [76] that the protocol under consideration must provide to the users of that protocol. The service specification should include properties of the protocol such as absence of deadlock and livelock, as well as specifying the allowable sequences of events between the protocol and its users.
3.1. Overview of the Protocol Verification Methodology

A key concept in the definition of the service specification is the notion of a service primitive. A service primitive represents an interaction between the user of the service (often another protocol entity in a higher layer) and the provider of the service [17] at the interface between protocol and user. This interaction may be, for example, a request by a user to establish a connection or an indication of the reception of data to the user. The set of allowable sequences of service primitives at the interface between the protocol and its users is known as the service language.

In this thesis, the service of the Stop-and-Wait Protocol is not complicated. A formal specification can be generated by hand, as a FSA and regular expression as is done in Chapter 5. However, in many situations (e.g. [60, 65, 100, 111, 148]), a formal specification of the service is too complicated to be generated by hand. In such cases, formal models must be used to represent the service specification. Coloured Petri nets, for example, can be used for this purpose. In essence, the transitions will usually be labelled by service primitives and so all sequences of service primitives are captured in the reachability graph. This reachability graph can then be mapped to a Finite State Automaton that encodes the service language, by designating a set of halt states [143]. Any non-primitive transition label is mapped to epsilon in the resulting FSA and service language.

3.1.2 Protocol Specification

The second task is to specify a protocol entity for each party involved in communication [17]. Whereas the service specification defines what a protocol should do, the protocol specification defines how the protocol should do it. In this thesis, Coloured Petri nets are used for this purpose. There are two entities involved in communication in the SWP, namely the sending entity (the Sender) and the receiving entity (the Receiver).

It is common practice [17] to also consider the underlying service at this step. This is the service provided by the communication medium over which the protocol entities operate; the service provided by the underlying layers in the layered protocol architecture. This is combined with the protocol specification model to obtain a composite specification of the protocol entities communicating over the underlying service. Usually the underlying service is incorporated directly into the protocol specification. This is the case in this thesis and thus for simplicity we will simply refer to the composite specification as the protocol specification.

In a similar way to that described for the service language, the protocol language can be derived from the protocol specification. This specifies all sequences of user observable events (i.e. service primitives) that are exhibited by the protocol. Again, the reachability graph of the protocol model is interpreted as a FSA, where all non-primitive actions (representing internal protocol actions that are not visible to the users of the protocol) are mapped to epsilon, and all other protocol actions are mapped to service
primitives. Designation of halt states, which may not be a trivial task [17], is undertaken as before.

### 3.1.3 Analysis

The third step [17] is to analyse the protocol specification using reachability analysis and/or theorem proving to investigate the desired properties of the system. As already stated, we are considering reachability analysis (and not theorem proving) in this thesis.

Properties that may be investigated include general properties such as absence of deadlock and absence of livelock, and more specific protocol properties such as bounds on the number of messages in the underlying communication medium. For the SWP, these are specified in Chapter 5 and verified in Chapters 9 and 10. We briefly discuss some of the general and specific properties of interest and methods for verifying them. This is based on material from [17].

Deadlocks can be determined by examining the dead markings of the reachability graph of the protocol specification [17]. These are dead markings that are undesired in some way. For example, a communication protocol between two entities that terminates with one entity in a closed state and the other entity in an open state. This is opposed to desired terminal states, also dead markings, which represent correct (and usually expected) terminal states of the system, often known a priori. For example, a communication protocol that terminates with both entities in a closed state.

Livelocks are cycles of repeated behaviour that once entered, cannot be escaped, and within which no progress is made with respect to the aim of the protocol. Such livelocks can be detected by calculating the strongly connected components (SCC) of the reachability graph. SCCs are mutually exclusive maximal subsets of mutually reachable states [81]. Each terminal SCC can be examined to determine whether it is a dead marking or a component involving cyclic behaviour. If the cyclic behaviour is not desired, then it is usually considered a livelock.

As an example of a protocol specific property, the maximum bounds on the number of messages and acknowledgements in the channels are also of interest. Such information can be used for buffer and network dimensioning, or may highlight the need for implementation of congestion control procedures.

### 3.1.4 Comparison

The fourth step is to compare the service specification with the protocol specification to see if the protocol specification is a faithful refinement of the service [17].

Given that deadlocks and livelocks can be detected directly from the reachability graph, this step is usually only concerned with comparing sequences of user observable events, i.e. comparing the service and protocol languages. In this thesis, the notion of language equivalence from Section 2.2.1 is used for this comparison. A similar notion that may be used, although slightly weaker, is language inclusion.
3.2 Summary and Limitations

![Diagram of Protocol Verification Methodology](image)

Figure 3.1: Protocol verification methodology

(Also from Section 2.2.1).

If the protocol language is the same as the service language (i.e. language equivalence holds) then the protocol is considered a faithful refinement of the service [17]. If the protocol language is a subset of the service language (i.e. language inclusion holds) then in some cases, the protocol may also be a faithful refinement of the service, implementing an acceptable subset of the service only. What constitutes an acceptable subset of the service is very much protocol-specific and is beyond the scope of this thesis. The interested reader is directed to [111, 112] for an example involving an electronic commerce protocol, the Internet Open Trading Protocol (IOTP) [27, 28].

However, if the protocol language is not contained in the service language, then the offending sequences of events represent behaviour in the protocol that is not specified or allowed by the service. This may indicate an error in the protocol model or in the protocol itself.

### 3.2 Summary and Limitations

The protocol verification methodology as applied to CPNs can be summarised in Fig. 3.1, taken from [17, 100]. The procedures for verifying properties of a protocol are shown in the dashed box on the right. The protocol definition, often from an international standard, is modelled using CPNs, resulting in the protocol CPN. The reachability graph of this protocol CPN can be generated using software tools such as Design/CPN [39] or CPN Tools [38]. This reachability graph can then be analysed to determine deadlocks, livelocks and other dynamic behaviour of the protocol.

The steps required to verify a protocol against its service are shown in the dashed box on the left of Fig. 3.1. From a service definition, again often from an international standard, a service CPN is created from which the reachability graph can be generated. As described above, the service language can be captured by treating the reachability graph as a FSA and applying standard FSA reduction techniques as described in Chapter 2. (As mentioned in Section 3.1.1 the service language of the Stop-and-Wait proto-
3.3 Concluding Remarks

The Protocol Verification Methodology has been successfully applied to many real-world examples in the concrete domain, although we note that it has some limitations when it comes to parametric verification of protocols. In the next chapter, we discuss attempts (both concrete and parametric) to verify the Stop-and-Wait Protocol and close variants.
Chapter 4

Previous Work

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, 25, 79, 94, 95, 107, 123, 131] and [119, 156] for surveys of techniques, and a parameter that influences the operation of an entity or a fixed number of entities. In this thesis, ‘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 4.1 introduces relevant literature on the verification of the Stop-and-Wait and Alternating Bit protocols in a non-parametric setting. Sections 4.2 and 4.3 describe and discuss two techniques applied to the parametric verification of the Stop-and-Wait Protocol. The approach described in Section 4.3 has been applied to the Sliding Window Protocol, which we discuss in Section 4.4. Previous work on developing algebraic expressions for parametric verification of other systems is discussed in Section 4.5. Finally, our own previous work is discussed in Section 4.6.

4.1 Non-Symbolic Verification of SWP

The simplest Stop-and-Wait Protocol (SWP) [127, 135] 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, 22, 40, 122, 129, 133–135, 139]. 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 [105] 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 [129] 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) [74]. 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 [133] and [134] 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 [139] 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 [26], LOTOS (Language Of Temporal Ordering Specifications) [24] and SDL [74]. 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 [22] 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 [30]) 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.
4.2. Symbolic Analysis using TReX

Reisig [122] 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 [40], and again, no analysis is performed.

Tanenbaum [135] 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.

4.2 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, 136] 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 [29]. 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 this thesis, 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 \texttt{MaxRetrans}. We also model an arbitrary maximum sequence number as a parameter, rather than being limited to a maximum sequence number of 1.
4.3 A Compositional Behavioural Fixed Point Approach

Valmari and his co-workers (e.g. [142,144,145]) use a behavioural fixed point method and compositional techniques for the verification of parametric systems. In [144] a variant of the ABP using limited retransmission, i.e. where there is an arbitrary bound (e.g. $\text{MaxRetrans}$) on the number of retransmissions, is verified using Chaos-Free Failures Divergences (CFFD) equivalence [143, 145]. The behavioural fixed point method allows the modelling of $\text{MaxRetrans}$ 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.

There are several differences with the work presented in this thesis (taken from [19,20]). Perhaps the most significant is that the channels are limited to a capacity of one, whereas the channels considered in this thesis are unbounded. Valmari [144] 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 5 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.

4.4 The Sliding Window Protocol

Previous work in a similar vein involves verification of the Sliding Window protocol [130], 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 [85, 86] examines the Sliding Window protocol for various window sizes, unbounded retransmissions, arbitrary channel capacity and modulo (wrapping) sequence numbers. The compositional technique from [142, 144, 145] 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 [124, 152] 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, this thesis considers 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 [88], referenced by Kaivola [85], 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, this thesis 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.

4.5 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 [21, 65]. The Data Transfer service of 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 the work in this thesis 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 this thesis by my colleague Lin Liu [100, 102], 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 [78], *Control protocol for multimedia communications*. In [100, 102], 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 [100]. In the lossy case, however, when mapping to a recursively defined FSA, some transitions map to $\epsilon$, 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 $\epsilon$-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 $\epsilon$ 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 [100] and the work in this thesis is that, for the lossy service case, [100] develops parametric reachability graphs and finite state automata that are defined recursively, whereas in this thesis the parametric reachability graph and finite state automaton developed in Chapters 8 and 10 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 determinisation through the use of epsilon closures (as described in Chapter 2). Another key difference is that, as with the work in [65], the work in [100] 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, the work in this thesis involves two protocol parameters, neither of which are the single medium capacity parameter.

4.6 Our own Previous Work

This section summarises work that has been undertaken with my supervisor (and occasionally other colleagues) that is not included in this thesis, except [50, 51] (and to a lesser extent [48, 49]) which are included.

In our previous work [17] we summarised a protocol verification methodology based on CPNs [81]...
and finite state automata. This methodology uses state space methods and has been applied successfully for finite state systems, for small values of parameters. A brief overview of this methodology was presented in Chapter 3. Techniques such as partial orders, symmetry and equivalence, and packed state spaces [42, 115, 143], and more recently the sweep-line method [18, 32, 91, 106], for alleviating the state space explosion problem [143] help to extend the method to larger ranges of parameters, but cannot handle large or unbounded values.

In [16,17], the methodology is illustrated using a Stop-and-Wait Protocol model with two parameters: the maximum sequence number, \( \text{MaxSeqNo} \); and the maximum number of retransmissions, \( \text{MaxRetrans} \). This model resembles the model presented in Section 5.5. 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 [17] 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 \( \text{MaxSeqNo} \), and any non-negative value of \( \text{MaxRetrans} \). For reordering channels, a hand proof of the unboundedness of the message and acknowledgement channels was developed in [15–17] using the semantic model of High-level Petri Nets [73]. For other properties, such as that the protocol conforms to its service of alternating send and receive events (described in Section 5.2), 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 [17] or lossy channels [15,17].

In [19, 20] we have used a symbolic verification tool called FAST (Fast Acceleration of Symbolic Transition systems) [44] 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, 45]. In [20] a methodology was described for converting CPNs into counter systems and the SWP CPN model from [15, 17] 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 \( \text{MaxRetrans} \) and for \( \text{MaxSeqNo} \) from 1 to 5 in [19, 20], with the results extended to arbitrary \( \text{MaxSeqNo} \) but with \( \text{MaxRetrans} = 0 \) in [19]. 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 [17] for this range of parameter values. A further result presented in [19] was an automatic proof of the form of the markings of the parametric reachability graph (the algebraic expressions for the markings) as presented in [48, 49], in $\text{MaxSeqNo}$ only, with $\text{MaxRetrans}=0$. This complemented the manual proof already performed in [48, 49].

Progress was made toward parametric verification of the Stop-and-Wait Protocol (prior to and concurrently with [19, 20]) in [48, 49], when considering the SWP with no retransmissions. Algebraic expressions representing the family of reachability graphs were developed in the $\text{MaxSeqNo}$ parameter only, (i.e. with $\text{MaxRetrans}=0$). These expressions for the parametric reachability graph in $\text{MaxSeqNo}$ were then used to parametrically verify the conformance of the SWP to its service of alternating send and receive events, for all values of $\text{MaxSeqNo}$.

Further progress was made in [50, 51], in which some of the core results of this thesis are presented. The algebraic expressions for the parametric reachability graph of the SWP were extended to incorporate both the $\text{MaxSeqNo}$ and $\text{MaxRetrans}$ parameters. Incorporation of the $\text{MaxRetrans}$ parameter increased the complexity of the expressions considerably. As discussed in Section 6.2, 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 $\text{MaxSeqNo}$ parameter but quartically in the $\text{MaxRetrans}$ parameter. Verification of a number of properties from the parametric reachability graph (see Chapter 9) are sketched in [50] and carried out in [51]. However, neither [50] nor [51] prove conformance of the SWP to its service of alternating send and receive events in both parameters, as we do in Chapter 10.

### 4.7 Concluding Remarks

A significant body of literature exists in relation to the verification of the Stop-and-Wait Protocol and variants such as the Alternating Bit, Bounded Retransmission and Sliding Window protocols, carried out in the concrete (non-symbolic) and parametric domain, both by ourselves and by others. There have also been significant results relating to the development of algebraic expressions (in a single parameter, the medium capacity) for the parametric verification of other protocols. However, this thesis is the first to develop algebraic expressions in two parameters for the reachability graph of the parameterised Stop-and-Wait Protocol, in order to verify this infinite class of systems for every possible value of the maximum sequence number and maximum number of retransmissions parameters. Next, as the first step toward achieving this parametric verification, the Stop-and-Wait Protocol as treated in this thesis is described, modelled, and properties for verification are defined.
Chapter 5

Stop-and-Wait Flow Control Protocol

Stop-and-Wait is an elementary form of flow control [127, 135] between a sender and a receiver. In this chapter, we begin by presenting an introduction to the Stop-and-Wait Protocol in Section 5.1, along with a narrative description of the SWP as modelled in this thesis. This is followed by a formal definition of the service that the SWP should provide to its users in Section 5.2. Sections 5.3 and 5.4 present the modelling assumptions we have made with respect to our SWP CPN model and the parameterisation of this model, which is presented in Section 5.5 and formalised as a High-level Petri Net Graph in Section 5.6. Finally, Section 5.7 presents the properties that we wish to verify.

5.1 Introduction to the Stop-and-Wait Protocol

As the name suggests, a Stop-and-Wait Protocol can be considered to be any data transfer protocol in which the sending entity stops after transmitting a message and waits until it receives an acknowledgement indicating that the receiver is ready to receive the next message. The Stop-and-Wait Protocol is a data transfer protocol, and hence does not include any connection management procedures (e.g. establishment and tear-down of connections).

As mentioned in Section 4.3, we do not consider explicit communication with the users, but instead we rely on the send and non-duplicate receive actions to be considered as synchronised communication with the user. We do not consider the reporting of errors to the user.

We consider that the sender and receiver entities can each be in one of two states. For the sender, this is one state in which the sender is ready to send a new message, and another in which the sender is waiting for an acknowledgement of the currently outstanding message. For the receiver, this is one state in which it is ready to receive a message, and another in which it is processing a message and generating the appropriate acknowledgement with which to reply. Both the sender and receiver will alternate between their two respective states as protocol execution proceeds.
5.1. Introduction to the Stop-and-Wait Protocol

Both the sender and the receiver maintain a sequence number. In the case of the sender, this sequence number (the *sender sequence number*) records the sequence number of the message to send next, or if a message is currently outstanding, the sequence number of the message that is currently outstanding. In the case of the receiver, this sequence number (the *receiver sequence number*) records the sequence number of the next message expected by the receiver.

The basic operation of the sender and receiver protocol entities can be described as follows. Both the sender and receiver start with a sequence number of 0, so that the first message sent by the sender will have sequence number 0, and the first message expected by the receiver is a message with sequence number 0. When the sender sends a message, it enters a state in which it can not send another new message until it receives an acknowledgement of the message it just sent (the *currently outstanding message*). When the receiver receives this message, the receiver checks the sequence number of this message against the sequence number it is expecting. If they match, this means that the message received is the correct message, and the receiver will increment its sequence number to reflect that it is now expecting the ‘next’ message. The receiver responds to the sender with an acknowledgement containing the next sequence number expected. Once the sender receives this acknowledgement, compares its own sequence number with that received in the acknowledgement. If the received sequence number is one greater than the sequence number of the currently outstanding message, the sender knows that the receiver has successfully received the message. The sender increments its sequence number and can now proceed to send the next message.

Stop-and-Wait Protocols often operate over noisy channels and combine flow control with error recovery using a timeout and retransmission scheme, known as Automatic Repeat ReQuest (ARQ) [135]. When a message is sent, a timer is started, which will expire after some finite *timeout period*. A checksum [135] is included in the message to detect transmission errors. Messages that pass the checksum are acknowledged as received correctly. A message that fails the checksum is discarded by the receiver. Acknowledgements are also protected by a checksum. In the case of a checksum failure of either the message or acknowledgement, the sender of the message will not receive an acknowledgement before the expiration of the retransmission timer, and thus will retransmit the message. The retransmission will have the same sequence number as the original message.

Retransmissions may have two causes: transmission errors and delay. In the case of delay and acknowledgement transmission errors, retransmission will introduce unnecessary duplicates of the original message into the system. When the acknowledgement is not received by the sender before the timer expires, a retransmission will occur even though the first data message has been received correctly and, in the case of delay, also acknowledged correctly. Appending a sequence number to each message, as described above, prevents duplicate messages being accepted as new messages. In practice, however,
neither sequence numbers or the number of retransmissions are unbounded.

To circumvent the issue of unbounded sequence numbers in a practical system, the Stop-and-Wait Protocol has a maximum sequence number beyond which sequence numbers cannot increase. This defines a finite sequence number space ranging from 0 to the maximum sequence number. When the sequence number of the sender or receiver reaches this maximum value, and is incremented, the sequence number wraps back to 0. Hence, sequence numbers are incremented using modulo arithmetic, modulo the ‘maximum sequence number plus one.’

To prevent unbounded retransmissions, the Stop-and-Wait Protocol has a maximum number of retransmissions. The sender entity maintains a retransmission counter, which records the number of times the currently outstanding message has been retransmitted, and is reset to zero when the currently outstanding message is acknowledged. When the maximum number of retransmission attempts has been reached, and the outstanding message has still not been successfully acknowledged, then the sender gives up retransmitting on the assumption that a severe problem has occurred with the link (such as a cable being cut). This problem is dealt with by a management entity, and hence is not considered as part of the Stop-and-Wait Protocol itself.

Note that the receiver will send an acknowledgement regardless of whether the message received is the next expected message or a duplicate of a previously received message. This is because if only one acknowledgement is sent per new message, then the protocol will enter a deadlock if this one acknowledgement is lost.

5.1.1 Sender Procedures

The following narrative pseudo-code summarises the procedures of the Sender. The key points from these procedures are illustrated in Fig. 5.1.

1. Initial state:
   - Sender sequence number = 0
   - Retransmission counter = 0
   - Sender state = ready to send a message

2. The sender is in a state in which it is ready to send a message. Two things may happen while the sender is in this state:
   (a) the sender sends a message. In this case, the sender goes to step 3.
   (b) the sender receives an acknowledgement for a message. Because there are no messages currently outstanding, the sender discards this acknowledgement and takes no further action.
The sender returns to the start of step 2.

3. A Send Message event occurs. The sender:

(a) receives a message to send to the receiver from the sender user;
(b) sends this message, with its sequence number equal to the current sender sequence number, into the communication channel;
(c) places a copy of this message in a retransmission buffer;
(d) starts a retransmission timer; and
(e) changes state to be waiting for an acknowledgement.

The sender now moves on to step 4.

4. While the sender is waiting for an acknowledgement, three things may happen. Either:

(a) the retransmission timer expires because an acknowledgement of the currently outstanding message has not yet been received. In this case, the sender goes to step 5.
(b) the receiver receives an acknowledgement for the currently outstanding message. In this case, the sender goes to step 6.
(c) the receiver receives an old acknowledgement, i.e. an acknowledgement for a message that has already been successfully acknowledged. In this case, the sender discards the acknowledgement and takes no further action. The sender returns to the start of step 4.

5. If the number of retransmissions of the currently outstanding message has not yet reached the maximum, then this event triggers a retransmission, in which:

(a) a copy of the currently outstanding message, with the same sequence number as the original message, is sent into the communication channel;
(b) the retransmission counter is incremented by one;
(c) the retransmission timer is reset; and
(d) the sender remains in a state waiting for an acknowledgement. The sender returns to step 4.

If the number of retransmissions of the currently outstanding message has reached the maximum limit, then the sender gives up and notifies the user that there is a problem. This pseudo-code terminates.

6. When an acknowledgement of the currently outstanding message is received, the sender:

(a) stops the retransmission timer;
5.1. Introduction to the Stop-and-Wait Protocol

(b) resets the retransmission counter to 0;

c) removes the message from the retransmission buffer;

d) increments its sequence number, wrapping the sequence number back to 0 if the maximum sequence number has been reached; and

e) returns to a state in which it is ready to send a message.

The sender returns to step 2.

5.1.2 Receiver Procedures

The following narrative pseudo-code summarises the procedures of the Receiver. The key points from these procedures are illustrated in Fig. 5.2.

1. Initial state:

   - Receiver Sequence Number = 0
   - Receiver state = ready to receive a message

2. The receiver is in a state in which it is ready to receive a message. It is expecting a message with the next expected sequence number. Two things may happen while the receiver is in this state:

   a) the receiver receives a message with the expected sequence number. In this case, the receiver goes to step 3.

   b) the receiver receives a duplicate message (its sequence number does not equal the next expected sequence number). In this case, the duplicate message is discarded and the receiver goes to step 4.

3. A Receive Message event occurs. The receiver:

   - passes this message to the receiver user; and

   - increments its sequence number, wrapping the sequence number back to 0 if the maximum sequence number has been reached.

   The receiver now moves to step 4.

4. The receiver sends an acknowledgement, containing the next sequence number expected. The receiver:

   - sends this acknowledgement into the communication channel; and
5.1. Introduction to the Stop-and-Wait Protocol

1. Initial state:
   - Sender Sequence Number = 0.
   - Retransmission Counter = 0.

2. Sender is ready to send a message.
   - Receive and discard an acknowledgement.

3. A ‘Send Message’ event:
   - Send a copy of the message into the message channel.
   - Copy message to retransmission buffer.
   - Start retransmission timer.

4. Sender is waiting for the correct acknowledgement.
   - Receive and discard a duplicate acknowledgement.

5. If the retransmission counter has not reached its maximum:
   - Retransmit the message currently stored in the retransmission buffer.
   - Increment the retransmission counter.
   - Reset the retransmission timer.

6. If the correct acknowledgement is received:
   - Stop the timer.
   - Reset the retransmission counter to 0.
   - Clear the retransmission buffer.
   - Increment Sender Sequence Number (modulo).

Figure 5.1: Key procedures of the Sender.

1. Initial state:
   - Receiver Sequence Number = 0.

2. Receiver is ready to receive a message.

3. A ‘Receive Message’ event:
   - Pass this message to the receiver user.
   - Increment Receiver Sequence Number (modulo).

4. Send acknowledgement indicating the next expected message.
   - Receive and discard a duplicate message.

Receiver is generating an acknowledgement.

Figure 5.2: Key procedures of the Receiver.
returns to a state in which it is ready to receive a message.

The receiver now returns to step 2.

5.2 The Stop-and-Wait Service

Following the Protocol Verification Methodology presented in Chapter 3, the first step is to define the service [76] provided by the Stop-and-Wait protocol. Recall that the service defines a set of service primitives and their global sequences as observed by the SWP users. It should also specify all properties of the service that a SWP should provide to its users. This includes properties relating to the sequencing of user observable events as well as properties like absence of deadlock and livelock. As described in Chapter 3, properties like absence of deadlock and livelock are investigated using reachability analysis. These are formalised in Section 5.7.

5.2.1 Sequences of Events

For the SWP service, we define two service primitives: a Send at the sender entity interface; and a Receive at the receiver entity interface, corresponding to the sender user submitting an item of data to be sent and the receiver user receiving an item of data, respectively. Note that any internal protocol actions, which may include sending and receiving retransmissions and acknowledgements, are not visible to the user, and so are not represented by service primitives. The service language can now be defined. Recall that this is the set of all allowable global sequences of service primitives. It is often the case that a model of the service must be created in order to specify the service language, however the Stop-and-Wait service is simple enough to allow direct manual specification. The allowable sequences can be defined using the regular expression in Definition 5.1, illustrated by the FSA in Fig. 5.3.

**Definition 5.1 (Service Language).**

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

The service language specifies sequences of alternating send and receive events, which may end
5.2. The Stop-and-Wait Service

Figure 5.4: An FSA of an alternative Stop-and-Wait Service Language.

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, mentioned in Section 5.1, 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 \((\text{Send } \text{Receive})^{\ast}\), 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 thesis.

An interesting philosophical debate may be held as to whether the service specification should include the *empty* service, that is, whether or not the service language should include the empty sequence. Logically, it is reasonable to expect that for a service to have been used, there must have been at least one interaction with the user, i.e. the occurrence of at least one service primitive. An alternative, but slightly more complex, service language that does not include the empty sequence is presented in Definition 5.2 and illustrated by the FSA in Fig. 5.4.

**Definition 5.2 (Alternative Service Language).**

The *Stop-and-Wait Service Language* of alternating send and receive events, excluding the empty sequence of events (the empty service) is given by the regular expression

\[
\text{Send } (\text{Receive } \text{Send})^{\ast} \text{Receive}^{\dagger}
\]

where \(\text{Receive}^{\dagger}\) represents 0 or 1 repetitions of the **Receive** primitive.

Sequences in this language may also end with a **Send** primitive, but all sequences start with a **Send** primitive and thus have a length greater than 0. We do not consider this alternative service in this thesis, and hence the former definition, including the empty service, will be used. From a correctness perspective
5.2. The Stop-and-Wait Service

it does not matter whether the empty service is included in the service specification, and including the empty service actually simplifies our task, as discussed in the proof of Theorem 8.1 in Chapter 8.

5.2.2 Data Independence

Operation of the SWP is unaffected by the data it transmits. Accordingly, the service specification abstracts from data completely. This service specification is adequate for the purposes of verifying sequences of events. However, the abstraction is too strong to allow verification of properties concerning data.

In order to prove properties concerning data, data independence \([96, 124, 152]\) principles must be applied. It was shown in \([152]\) that, in essence, if a property of a system concerns \(n\) distinct values of data, then the model of that system must capture at least \(n + 1\) distinct data values. Or, conversely, that the large (conceptually infinite) set of data values over which the real system operates can be abstracted to \(n + 1\) data values in the model. (Note that properties concerning sequences of service primitives require 0 distinct data values, and that abstracting from data completely is equivalent to abstracting all data values to a single data value.)

To highlight why this may be useful, consider a Stop-and-Wait Protocol operating over a reordering channel (as investigated in \([15, 17]\)). We found that, with alternating bit sequence numbers, the protocol exhibited undesired sequences of send and receive events, with, for example, two receive events for every send event (implying data duplication) or three send events for each receive event (implying data loss). All of these erroneous sequences violate the property of alternating send and receive events, and so would be detected with the service language of Definition 5.1. However, closer investigation of the reachability graph of the corresponding model revealed that even some sequences that conform with this service language may include loss and duplication of data. Such sequences would thus remain undetected.

Hence, in order to prove properties about data, as done in e.g. \([85, 124]\), a more detailed service specification is needed. The main goal of this thesis is to present a novel formalism, based on algebraic expressions over explicit MaxSeqNo and MaxRetrans parameters, to represent the reachability graphs of the infinite class of Stop-and-Wait protocols, and to perform parametric verification of properties. The properties proved do not concern data. Thus, a complete abstraction from data is adequate for this purpose. Some initial investigations into both defining such an alternate service specification and verification of the SWP against this specification are described and discussed in Section 11.2.
5.3 Modelling Scope and Assumptions

A number of assumptions and simplifications are made about the Stop-and-Wait Protocol and its environment, in order to simplify the task of modelling and parametric verification.

**Abstraction 5.1** (Data Independence). *Data is ignored.*

From the discussion in Section 5.2.2, the SWP is independent of the data it transmits, and the properties we wish to verify do not relate to data. Hence, data does not need to be modelled. (All elements of data are abstracted to a single element of data, and hence does not need to be modelled.)

**Abstraction 5.2** (One Sender, One Receiver). *Only one sender and one receiver is considered.*

In a single session each message (acknowledgement) sent by the sender (receiver) will only ever have one possible destination. Hence, this is a generic representative of any session of the SWP and removes the need to model an addressing mechanism in both the sender and receiver entities. These two abstractions allow each message and acknowledgement to be modelled by just its sequence number.

**Abstraction 5.3** (Protocol Users). *The users of this protocol are not explicitly modelled.*

The users, and the explicit communication between user and protocol, do not need to be modelled. It suffices for the Send and non-duplicate Receive actions of the protocol to be considered as synchronised communication with the users. We assume that data is ready to be sent by the sender user each time a Send action occurs, and that data is passed to the receiver user each time a non-duplicate receive action occurs. As mentioned in Section 4.3, the reporting of errors to the sender user (e.g. [144]) or both users (e.g. [3]) is not considered.

**Abstraction 5.4** (Retransmission Timer). *The retransmission timer at the sender is modelled implicitly.*

We only need to model the event of the retransmission timer expiring, not the timer itself. Hence, expiry of the retransmission timer is modelled by the action that retransmits a message.

**Abstraction 5.5** (Channel Characteristics). *The underlying channel is lossy and in-order (FIFO), and has an unbounded capacity.*

In practice it may be difficult or impossible to determine a bound on the underlying channel capacity. By modelling the underlying channel with an unbounded capacity it is possible to determine whether a structural bound on the number of messages and acknowledgements is imposed by the protocol itself, and if so, what capacity the underlying channels must have in a practical implementation of the protocol.
Abstraction 5.6 (Corruption of Messages). Corruption of messages and acknowledgements can be modelled by loss.

It is assumed that the underlying medium may corrupt messages in a way that can be detected e.g. by a Cyclic Redundancy Check (CRC) [127]. Upon detection, corrupted messages (acknowledgements) are discarded by the receiver (sender), which corresponds to a loss of that message (acknowledgement). This allows corruption of messages to be implicitly modelled by the loss mechanism.

5.4 Parameterisation

We consider the Stop-and-Wait Protocol to have a finite sequence number space and an upper bound on the number of retransmission attempts at the sender. We therefore parameterise our CPN model with the Maximum Sequence Number, $\text{MaxSeqNo}$, and the Maximum Number of Retransmissions, $\text{MaxRetrans}$, respectively.

If we allow $\text{MaxSeqNo}$ and $\text{MaxRetrans}$ to be unbounded, an infinite class of SWP CPN models that require verification results. As illustrated in Chapter 6, conventional state space methods permit analysis of the system only for small to moderate values of the parameters due to state explosion [143].

In general, the use of parameters in modelling with CPNs (as with other languages) can provide many benefits. Importantly many different system configurations can be modelled and analysed without needing a separate model for each. Parameter values can be chosen to reflect the corresponding parameter values of the system being modelled, such as specifying $\text{MaxSeqNo} = 1$ to obtain the ABP. In other instances, it may not be possible to determine the value of a parameter from the system being modelled (it may even be unbounded) or it may be that constraints such as state space explosion [143] prevent investigation of the model for realistically large values of a system parameter. Different parameter values can be selected to analyse the system operating under different conditions. Some parameters range over a small set of values, such as the maximum value for the retransmission counter in the Transaction Layer of the Wireless Application Protocol (WAP) [60]. The value of this parameter never exceeds 8. Thus it is not necessary to verify correct protocol behaviour for values greater than 8. Other parameter values may be large or range over a large set of values, e.g. the maximum value of a sequence number. For example, the X.25 protocol [77] allows the use of 3 bit, 7 bit and 15 bit sequence numbers. TCP [120] uses 32 bit sequence numbers. In some cases the parameter value may not be static. For example, the Layer 2 Tunneling Protocol [138] specifies a configurable bound on the number of retransmission attempts. Also, as is often the case, the parameter values may simply be unknown or unspecified in the protocol documentation. For example, an upper bound on the retransmission of TCP segments is not specified in [120].
5.5. The Parameterised Stop-and-Wait CPN Model

The class of SWPs we are modelling is important because many practical data transfer protocols use a sliding window mechanism that has its foundation based on Stop-and-Wait principles. Sliding Window protocols [127, 135] improve the efficiency of SWPs by allowing many messages (rather than one) to be sent before requiring an acknowledgement. The number of messages that can be sent before the sender must stop and wait to receive an acknowledgement is known as the window. Cumulative acknowledgements and more sophisticated error retransmission schemes (such as Selective Reject) [127] can also improve efficiency. These schemes are used in many practical protocols such as TCP [120]. The underlying principles of ARQ used in sliding window protocols are the same as those used in SWPs, so that a window size of 1 corresponds to a Stop-and-Wait protocol. Thus it is essential that the stop-and-wait mechanisms work correctly if the more advanced protocols are also to be correct.

It is well known [135] that for sliding window protocols to work properly in detecting and discarding duplicates, the sequence number space needs to be one greater than the number of unacknowledged messages (the window). In the case of Stop-and-Wait protocols which have just one outstanding unacknowledged message, the sequence number space can be just two numbers, usually \( \{0,1\} \). A Stop-and-Wait protocol realised with a one bit sequence number is called an Alternating Bit Protocol [9], because the sequence number can be implemented using just one bit in the header of the message, and the sequence number value alternates between 0 and 1.

However, the infinite class of SWPs modelled in this thesis considers every possible values of the maximum sequence number. This takes us a step closer to sliding window protocols, where the window size is arbitrary, and hence the sequence number space (which must be at least one greater than the window size) is also arbitrary. It may be the case that SWPs with larger sequence number spaces can work correctly over media with a limited amount of re-ordering (see [88] and Section 4.4), but this is not considered here and is outside the scope of this thesis.

5.5 The Parameterised Stop-and-Wait CPN Model

Our parameterised CPN model of the SWP operating over an in-order medium is given in Figs. 5.5 and 5.6. Figure 5.5 presents the graphical representation of the system, while (as in the example in Chapter 2) Fig. 5.6 defines all the constants, sets and functions required and declares the types of the variables used in the annotations associated with the graphical representation.

The model comprises three main parts: the Sender (on the left), the Receiver (on the right) and an underlying bidirectional communication medium, Network, in the middle. The parameters, MaxSeqNo and MaxRetrans, can be seen on lines 1 and 2 of Fig. 5.6 respectively. Let us denote the parametric SWP CPN by \( CPN_{(MS,MR)} \), where \( MS = \text{MaxSeqNo} \) and \( MR = \text{MaxRetrans} \).
Figure 5.5: The CPN of the Stop-and-Wait Protocol operating over an in-order (FIFO) underlying medium.
5.5. The Parameterised Stop-and-Wait CPN Model

1 val MaxRetrans = 0;
2 val MaxSeqNo = 1;
3
4 color Sender = with s_ready | wait_ack;
5 color Receiver = with r_ready | process;
6 color Seq = int with 0..MaxSeqNo;
7 color RetransCounter = int with 0..MaxRetrans;
8 color Message = Seq;
9 color MessList = list Message;
10
11 var sn,rn : Seq;
12 var rc : RetransCounter;
13 var queue : MessList;
14
15 fun NextSeq(n) = if (n = MaxSeqNo) then 0 else n+1;
16 fun Contains([],sn) = false
17      | Contains(m::queue,sn) = if (sn=m) then true else Contains(queue,sn);
18 fun Loss(m::queue,sn) = if (sn=m) then queue else m::Loss(queue,sn);

Figure 5.6: Global Declarations for the Stop-and-Wait Protocol CPN in Fig. 5.5.

5.5.1 The Sender

The sender consists of three places, four transitions and their interconnecting arcs. Place sender_state models the two states of the sender (either ready to send a new data message or awaiting an acknowledgement) and is typed by the colour set Sender (line 4 of Fig. 5.6). It is given the initial marking of one s_ready token, indicating that the Sender is initially in the ready state. The send_seq_no place stores the sender sequence number. It is typed by the colour set Seq (sequence number, line 6 of Fig. 5.6) and has an initial marking of a single 0 token, indicating that the first message to be sent will have sequence number 0. The current number of retransmissions is recorded in place retrans_counter, typed by the colour set RetransCounter (line 7 of Fig. 5.6). It is initially 0.

Transition send_mess models the sending of a message to the receiver. As discussed in Section 5.3, a message (or an acknowledgement) can be modelled by just its sequence number (line 8 of Fig. 5.6). When the sender is ready (place sender_state contains a s_ready token) the send_mess transition may occur. It writes its sequence number (as the message) to the message channel and changes the sender state to be waiting for an acknowledgement. Details on the operation of the message channel are given in Section 5.5.3.
5.5. The Parameterised Stop-and-Wait CPN Model

The **timeout_retrans** transition models the expiry of the retransmission timer and the retransmission of the currently outstanding message. This transition can only occur if the sender is waiting for an acknowledgement (place `sender_state` contains a `wait_ack` token) and there have been less than `MaxRetrans` retransmissions of this message, i.e. the value of `rc` (line 10 of Fig. 5.6) is less than `MaxRetrans` (see the guard). When **timeout_retrans** occurs, the retransmission counter is incremented by 1 and the retransmitted message is written to the message channel.

Transition **receive_ack** models the receipt of expected acknowledgements from the receiver, i.e. those that acknowledge the currently outstanding message. Duplicate acknowledgements are received and discarded by transition **receive_dup_ack**. Recall that duplicate acknowledgements may result from acknowledged retransmissions, where delay rather than loss was the cause of the retransmission. The complementary guards on these transitions identify the acknowledgement as being expected (i.e. it has a sequence number one greater than the sender sequence number) or a duplicate. The function `NextSeq` is used to increment the sequence number modulo (`MaxSeqNo` + 1), as shown on line 15 of Fig. 5.6. An occurrence of **receive_ack** will remove the acknowledgement from the channel, return the sender to the ready state, reset the retransmission counter to 0 and increment the sequence number stored in `send_seq_no` using `NextSeq`. The transition **receive_dup_ack** discards duplicate acknowledgements irrespective of the state of the sender.

### 5.5.2 The Receiver

The receiver consists of two places and two transitions. The state of the receiver is modelled by the place `receiver_state`, either ready to receive a message or processing a message. It is typed by the colour set `Receiver` (line 5 of Fig. 5.6) and has an initial marking of `r_ready`. The `recv_seq_no` place records the sequence number of the message expected next by the receiver, and is typed by the colour set `Seq`. As in the sender, this place is initially marked with a 0 token, indicating that the receiver is expecting a message with sequence number 0.

Transition **receive_mess** models the receipt of a message from the sender. The annotation on the arc from `receive_mess` to `recv_seq_no` compares the sequence number of the message (`sn`, line 11 of Fig. 5.6) with the sequence number expected by the receiver (`rn`, line 11 of Fig. 5.6). If they match, then the receiver interprets this message as the message it is expecting, and the sequence number is incremented by the `NextSeq` function and placed in `recv_seq_no`. If `sn` and `rn` do not match, a duplicate is detected (and discarded) and the receiver sequence number in `recv_seq_no` remains unchanged. Transition **send_ack** sends an acknowledgement containing the next sequence number expected by the receiver and returns the receiver to the ready state.
5.5. The Parameterised Stop-and-Wait CPN Model

5.5.3 The Underlying Medium

The underlying communication medium is modelled as a bidirectional channel consisting of one place and one transition for each direction of communication. The mess\_channel place models the message channel while the ack\_channel place models the acknowledgement channel. The two transitions mess\_loss and ack\_loss model the loss of messages and acknowledgements respectively. This corresponds to either loss in the network (due to congestion and buffer overflow in a router), or to discarding messages (and acknowledgements) due to checksum failures.

An in-order underlying medium can be considered as two queues, one for messages and one for acknowledgements. In order to model the medium using queues, the colour set MessList is introduced on line 9 of the declarations in Fig. 5.6, representing a list of messages or acknowledgements. MessList is used to type the places mess\_channel and ack\_channel. The initial marking of these two places is an empty list, [], representing that initially there are no messages and no acknowledgements in the channels.

The mechanism for interacting with a medium represented by a list must be explained. Each channel place has arcs in both directions, to and from each transition it interacts with. For example, the mess\_channel place has arcs leading to and from each of the transitions send\_mess, timeout\_retrans, mess\_loss and receive\_mess. All arcs incident on the two channel places have inscriptions written to manipulate the corresponding lists as FIFO queues. The operator ^^ concatenates two lists, while :: (the cons operator) concatenates an element to the head of a list. (Both list concatenation and the cons operator are functions that are built in to the software tool.) To illustrate the operation of the FIFO queues, consider transition send\_mess. When sending a message (adding something to the queue), the variable queue (line 13 of Fig. 5.6) is bound to the list in place mess\_channel and is thus removed from mess\_channel when send\_mess occurs. The new message is concatenated to the end of this list (expression queue ^^ [sn]), and the list is returned to place mess\_channel. Now consider transition receive\_mess. When receiving a message (removing something from the queue), the variable sn is bound to the first element of the list and the variable queue is bound to the rest of the list. When receive\_mess occurs, the list is removed from place mess\_channel and the list minus the first element (the value of queue) is returned to mess\_channel. The acknowledgement channel behaves in an analogous way.

Loss of messages and acknowledgements is more complicated. Our model of loss allows for messages and acknowledgements to be lost from anywhere in the queue. This reflects the situation where loss may occur in the underlying medium due to e.g. router overflow, or by corruption and subsequent discarding of messages or acknowledgements by the corresponding recipient. The ML function Contains (lines 16 and 17 of Fig. 5.6) is the guard of transitions mess\_loss and ack\_loss. The variable queue is bound to the list of sequence numbers from the channel place and the variable sn is nondeterministically chosen. The function Contains takes both of these values and returns true if sn is contained
5.5. The Parameterised Stop-and-Wait CPN Model

Table 5.1: Alternative models of loss operating on a channel containing [0, 0, 0, 1, 1, 1].

<table>
<thead>
<tr>
<th>Loss from the front of the channel only</th>
<th>Loss from anywhere in the channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 1, 1, 1]</td>
<td>[0, 0, 0, 1, 1, 1]</td>
</tr>
<tr>
<td>[0, 0, 1, 1, 1]</td>
<td>[0, 1, 1, 1]</td>
</tr>
<tr>
<td>-</td>
<td>[0, 0, 0, 1, 1]</td>
</tr>
<tr>
<td>[0, 1, 1, 1]</td>
<td>[0, 0, 1, 1]</td>
</tr>
<tr>
<td>-</td>
<td>[0, 0, 0, 1]</td>
</tr>
<tr>
<td>[1, 1, 1]</td>
<td>[1, 1, 1]</td>
</tr>
<tr>
<td>-</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>-</td>
<td>[0, 0]</td>
</tr>
<tr>
<td>[1, 1]</td>
<td>[1]</td>
</tr>
<tr>
<td>-</td>
<td>[0]</td>
</tr>
<tr>
<td>[1]</td>
<td>[]</td>
</tr>
</tbody>
</table>

in the list. If so, function Loss (line 18 of Fig. 5.6) removes the first instance of $sn$ from the list stored in queue. The correctness of this mechanism relies on the fact that the content of both the message and acknowledgement channels is always contiguous, so that the loss of the first instance of a sequence number is equivalent to loss of any instance of the same sequence number. As will be seen in Chapter 8, it turns out that this is the case for all reachable markings. The channel can be made lossless by replacing the guard of the two loss transitions with [false].

This is a more general model of loss than simply loss from the head of the queue, which can be modelled easily with the cons operator (::). It can be argued that the two models of loss are equivalent with respect to the behaviour of the protocol, however our chosen model of loss more closely reflects the real world when considering the protocol operating over a network such as the Internet. Surprisingly, our more complicated model of loss actually simplifies the representation of the underlying medium in Chapter 8. For example, consider messages being lost from a channel containing the messages [0,0,0,1,1,1] (the front of the channel is the left). Column 1 of Table 5.1 lists the possible content of the channel if
5.5. The Parameterised Stop-and-Wait CPN Model

loss can occur from the front of the channel only, and column 2 lists the additional configurations that are possible if loss can occur from anywhere. All combinations of zero to three instances of both 0 and 1 are present in column 2 (while respecting the FIFO order of these messages). We find it easier to represent symbolically the possibilities in column 2, rather than the restricted (in some sense ‘incomplete’) set of possibilities in column 1.

5.5.4 Discussion of Modelling Decisions

This model first appeared in [15–17], with in-order, reordering and fixed-capacity reordering channels. After a number of cosmetic changes, it attained its current form, which has also appeared in [19, 20, 48, 49]. Apart from the model of the channel, there are three cosmetic differences between the original model and the model presented here:

1. Explicit representation of receiver sequence number by a separate place, to make it consistent with the modelling style of the sender;
2. Folding of the sender state places onto a single place typed by the states of the sender; and
3. Folding of the receiver state places onto a single place typed by the states of the receiver.

Some of these changes, and related modelling decisions, are discussed below.

A straightforward way to begin modelling a system such as this is to use one place for each state of an entity, one place for each data item, and one transition for each action. This was the approach taken when modelling the sender in earlier versions of the SWP CPN model [15–17]. However, this is normally only possible for protocols with very few states. As the number of states increases, so do the number of arcs, which leads to a visually complex diagram with many arc crossings, distracting from the major flows. In order to simplify the SWP CPN diagram in Fig. 5.5, for example, the sender states and receiver states were folded and represented by single places typed by the set of states of the sender and receiver respectively, and thicker arcs were used to represent the control flow.

The transition receive\_mess represents both the receipt of an expected message and the discarding of duplicate messages, requiring a complex arc annotation. This is in contrast to the sender, where the receipt of acknowledgements is modelled using two transitions. This provides a more compact representation of the receiver at the expense of a small amount of visualisation of data flow.

The model structure aims to reflect the structure of the real life system, given a certain amount of abstraction. For example, the sequence numbers at the sender and receiver have been modelled in the net structure explicitly, in separate places. This is to reflect the fact that in an implementation, the sequence numbers as entities of data may exist separately from, and regardless of, the state of the sender or receiver. It also simplifies modelling of the sender, because duplicate acknowledgement messages can
be received and discarded regardless of the state of the sender. Conversely, the meaning of the value of
the sequence number in the sender (either the next to be sent when in state \textit{s.ready}, or the message to
be acknowledged when in state \textit{wait.ack}) is dependent on the state, and hence this would favour folding
the sequence number into the state places. Thus various trade-offs present themselves to the modeller,
even in a model as simple as this.

It may seem unusual that the sender does not increment its sequence number immediately upon
sending a message, which, if it did, would keep the sender sequence number always equal to the sequence
number of the next message to send. From the point of view of the operation of the Stop-and-Wait
Protocol, it makes no difference whether the sequence number of the sender is incremented as soon as
a message is sent or when the acknowledgement is received (provided, of course, that the inscriptions
are modified appropriately). However, the scheme that we have used allows for a simpler model, as the
sequence number of any retransmission of a currently outstanding message will be equal to the sender
sequence number. Thus the sequence number to retransmit does not need to be explicitly stored or
calculated each time from the sender sequence number.

5.6 The Parameterised Stop-and-Wait High-level Petri Net Graph

As mentioned in Chapter 2, a CPN is a form of High-level Petri Net. In general, CPNs and HLPNGs are
very similar, and will have the same semantics, so we can choose whichever representation best suits our
purposes for the task in hand. For example, we shall use the formal definition of a HLPNG to establish
behavioural properties of our SWP model in Chapter 6 and for derivation of notation in Chapter 7. We
now present a High-level Petri Net Graph representation of our SWP CPN model and discuss some minor
differences in representation.

A High-level Petri Net Graph representation, $HLPNG_{(MS,MR)}$, of our SWP CPN model is given
by $HLPNG_{(MS,MR)} = (NG, Sig, V ars, Alg, Type, AN, M_0)$ where:

- $NG = (P, T; F)$, where
  - $P = \{\text{sender.state, send.seq.no, retrans.counter, receiver.state, recv.seq.no, mess.channel, ack.channel}\}$;
  - $T = \{\text{send.mess, timeout.retrans, receive.mess, send.ack, receive.ack, receive.dup.ack, mess.loss, ack.loss}\}$; and
  - $F = \{(\text{sender.state, send.mess}), (\text{send.mess, sender.state}), (\text{send.seq.no, send.mess}), (\text{send.mess, send.seq.no}), (\text{mess.channel, send.mess}), (\text{send.mess, mess.channel}), \ldots \}$.
5.6. The Parameterised Stop-and-Wait High-level Petri Net Graph

\[(\text{sender\_state}, \text{timeout\_retrans}), (\text{timeout\_retrans}, \text{sender\_state}),\]
\[(\text{send\_seq\_no}, \text{timeout\_retrans}), (\text{timeout\_retrans}, \text{send\_seq\_no}),\]
\[(\text{retrans\_counter}, \text{timeout\_retrans}), (\text{timeout\_retrans}, \text{retrans\_counter}),\]
\[(\text{mess\_channel}, \text{timeout\_retrans}), (\text{timeout\_retrans}, \text{mess\_channel}),\]
\[(\text{mess\_channel}, \text{receive\_mess}), (\text{receive\_mess}, \text{mess\_channel}),\]
\[(\text{receiver\_state}, \text{receive\_mess}), (\text{receive\_mess}, \text{receiver\_state}),\]
\[(\text{recv\_seq\_no}, \text{receive\_mess}), (\text{receive\_mess}, \text{recv\_seq\_no}),\]
\[(\text{receiver\_state}, \text{send\_ack}), (\text{send\_ack}, \text{receiver\_state}),\]
\[(\text{recv\_seq\_no}, \text{send\_ack}), (\text{send\_ack}, \text{recv\_seq\_no}),\]
\[(\text{ack\_channel}, \text{send\_ack}), (\text{send\_ack}, \text{ack\_channel}),\]
\[(\text{ack\_channel}, \text{receive\_ack}), (\text{receive\_ack}, \text{ack\_channel}),\]
\[(\text{retrans\_counter}, \text{receive\_ack}), (\text{receive\_ack}, \text{retrans\_counter}),\]
\[(\text{send\_seq\_no}, \text{receive\_ack}), (\text{receive\_ack}, \text{send\_seq\_no}),\]
\[(\text{sender\_state}, \text{receive\_ack}), (\text{receive\_ack}, \text{sender\_state}),\]
\[(\text{ack\_channel}, \text{receive\_dup\_ack}), (\text{receive\_dup\_ack}, \text{ack\_channel}),\]
\[(\text{send\_seq\_no}, \text{receive\_dup\_ack}), (\text{receive\_dup\_ack}, \text{send\_seq\_no}),\]
\[(\text{mess\_channel}, \text{mess\_loss}), (\text{mess\_loss}, \text{mess\_channel}),\]
\[(\text{ack\_channel}, \text{ack\_loss}), (\text{ack\_loss}, \text{ack\_channel})\}.

\textbf{Sig} = (\text{Sorts}, \text{Ops}), \text{where}

- \text{Sorts} = \{\text{sender, receiver, retranscounter, seq, messlist, Bool}\}; \text{and}

- \text{Ops} = \{\langle (\text{messlist, messlist, messlist}), \langle (\text{seq, messlist, messlist}), \text{MakeList}(\text{seq, messlist}),\]
\[\text{Contains}(\text{messlist, seq, Bool}), \text{Loss}(\text{messlist, seq, messlist}), \text{NextSeq}(\text{seq, seq}),\]
\[\text{RetransEnabled}(\text{retranscounter, Bool}), \text{IncrementRC}(\text{retranscounter, retranscounter}),\]
\[\text{IncrementRSN}(\text{seq, seq}), \langle (\text{seq, Bool}), \langle (\text{seq, Bool}), \text{s\_ready}(\text{sender}),\]
\[\text{wait\_ack}(\text{sender}), \text{r\_ready}(\text{receiver}), \text{process}(\text{receiver}), 0, \text{retranscounter}, \text{true}\text{Bool}\}.

\textbf{Vars} = \{\text{sn}_\text{seq}, \text{rn}_\text{seq}, \text{queue}_\text{messlist}, \text{rc}_\text{retranscounter}\}

\textbf{Alg} = (\text{Types}, \text{Funs}), \text{where}

- \text{Types} = \{\text{Sender, Receiver, RetransCounter, Seq, MessList, BOOL}\} \text{where}

  * \text{Sender} = \{\text{s\_ready, wait\_ack}\};

  * \text{Receiver} = \{\text{r\_ready, process}\};

  * \text{RetransCounter} = \{0, 1, ..., MR\};

  * \text{Seq} = \{0, 1, ..., MS\};
5.6. The Parameterised Stop-and-Wait High-level Petri Net Graph

- MessList = Seq*, strings of 0 or more sequence numbers, where we represent strings of sequence numbers by a list of comma-separated elements enclosed by square brackets, for consistency with the CPN representation of lists; and
- BOOL = \{true, false\}.

- Funs = \{\^, ::, MakeList, Contains, Loss, NextSeq, RetransEnabled, IncrementRC, IncrementRSN, =, <>, s\ready, w\ack, r\ready, process, 0, true\},

where

- ^ : MessList × MessList → MessList is string (list) concatenation, so that ^([i_1, i_2, ..., i_m], [j_1, j_2, ..., j_n]) = [i_1, i_2, ..., i_m, j_1, j_2, ..., j_n];
- :: : Seq × MessList → MessList concatenates a sequence number to the start of a string (list) of sequence numbers, so that :: ([i_1, i_2, ..., i_m]) = [i_1, i_2, ..., i_m];
- MakeList : Seq → MessList converts a sequence number to a list containing only that sequence number, so that MakeList(i) = [i];
- Contains : MessList × Seq → BOOL returns true if the list contains the sequence number, so that Contains([i_1, i_2, ..., i_m], j) = \{true, if j ∈ \{i_1, i_2, ..., i_m\},
false, if j ∉ \{i_1, i_2, ..., i_m\}\}.
- Loss : MessList × Seq → MessList removes an instance of a sequence number from a list, so that Loss([i_1, i_2, ..., i_m, j, k_1, k_2, ..., k_n], l) = [i_1, i_2, ..., i_m, k_1, k_2, ..., k_n] and j is the first instance of l appearing in the list;
- NextSeq : Seq → Seq increments a sequence number, so that
NextSeq(i) = \{0, if i = MaxSeqNo,
i + 1, otherwise.\}
- RetransEnabled : RetransCounter → BOOL returns true if the retransmission counter has not yet reached MaxRetrans,
so that RetransEnabled(c) = \{true, if c < MR,
false, otherwise.\}
- IncrementRC : RetransCounter → RetransCounter increments the retransmission counter, so that IncrementRC(c) = c + 1 only for c < MR;
- IncrementRSN : Seq × Seq → Seq increments the receiver sequence number only when the expected message is received, so that
IncrementRSN(sn, rn) = \{NextSeq(rn), if sn = rn,
rn, otherwise.\}
5.6. The Parameterised Stop-and-Wait High-level Petri Net Graph

* $= : \text{Seq} \times \text{Seq} \rightarrow \text{BOOL}$ returns true if two sequence numbers are the same, so that $= (a, b)$ equals

\[
\begin{cases} 
\text{true, if } a = b, \\
\text{false otherwise.}
\end{cases}
\]

* $<> : \text{Seq} \times \text{Seq} \rightarrow \text{BOOL}$ returns true if two sequence numbers are not the same, so that $<> (a, b)$ equals

\[
\begin{cases} 
\text{true, if } a \neq b, \\
\text{false, otherwise.}
\end{cases}
\]

* $s\text{ready} \in \text{Sender}$ and $\text{wait}_\text{ack} \in \text{Sender}$ are the two states of the sender;

* $r\text{ready} \in \text{Receiver}$ and $\text{process} \in \text{Receiver}$ are the two states of the receiver;

* $0 \in \text{RetransCounter}$ is the integer value “zero”; and

* $\text{true} \in \text{BOOL}$ is the boolean value “true”.

\[
\begin{align*}
\text{Type}(p) &= \begin{cases} 
\text{Sender, if } p = \text{sender\_state}, \\
\text{Receiver, if } p = \text{receiver\_state}, \\
\text{Seq, if } p \in \{\text{send\_seq\_no, recv\_seq\_no}\}, \\
\text{RetransCounter, if } p = \text{retrans\_counter}, \\
\text{MessList, if } p \in \{\text{mess\_channel, ack\_channel}\}.
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\text{AN} &= (\text{ATerm}, \text{TCond}), \text{ where}
\end{align*}
\]

- $\text{ATerm}(u, v) =$

  * $s\text{ready}_{\text{sender}}$, if $(u, v) \in \{(\text{sender\_state, send\_mess}), \\
  (\text{receive\_ack, sender\_state})\}$;

  * $\text{wait}_\text{ack}_{\text{sender}}$, if $(u, v) \in \{(\text{send\_mess, sender\_state}), \\
  (\text{sender\_state, receive\_ack}), (\text{sender\_state, timeout\_retrans}), \\
  (\text{timeout\_retrans, sender\_state})\}$;

  * $r\text{ready}_{\text{receiver}}$, if $(u, v) \in \{(\text{receiver\_state, receive\_mess}), \\
  (\text{send\_ack, receiver\_state})\}$;

  * $\text{process}_{\text{receiver}}$, if $(u, v) \in \{(\text{receiver\_state, send\_ack}), \\
  (\text{receive\_mess, receiver\_state})\}$;

  * $sn\text{seq}$, if $(u, v) \in \{(\text{send\_seq\_no, send\_mess}), \\
  (\text{send\_mess, send\_seq\_no}), (\text{send\_seq\_no, receive\_ack}), \\
  (\text{send\_seq\_no, timeout\_retrans}), (\text{timeout\_retrans, send\_seq\_no}), \\
  (\text{send\_seq\_no, receive\_dup\_ack}), (\text{receive\_dup\_ack, send\_seq\_no})\}$;

  * $rn\text{seq}$, if $(u, v) \in \{(\text{recv\_seq\_no, send\_ack}), (\text{send\_ack, recv\_seq\_no}), \\
  (\text{recv\_seq\_no, receive\_mess})\}$;
5.6. The Parameterised Stop-and-Wait High-level Petri Net Graph

\[
\begin{align*}
\ast \text{queue}_{\text{messlist}}, & \text{ if } (u, v) \in \{(\text{mess}_\text{channel}, \text{send}_\text{mess}), \\
& (\text{receive}_\text{mess}, \text{mess}_\text{channel}), (\text{ack}_\text{channel}, \text{send}_\text{ack}), \\
& (\text{receive}_\text{ack}, \text{ack}_\text{channel}), (\text{mess}_\text{channel}, \text{timeout}_\text{retrans}), \\
& (\text{receive}_\text{dup}_\text{ack}, \text{ack}_\text{channel}), (\text{mess}_\text{channel}, \text{mess}_\text{loss}),
\end{align*}
\]

\[
\begin{align*}
(\text{ack}_\text{channel}, \text{ack}_\text{loss})\};
\ast \text{rc}_{\text{retranscounter}}, & \text{ if } (u, v) \in \{(\text{retrans}_\text{counter}, \text{receive}_\text{ack}), \\
& (\text{retrans}_\text{counter}, \text{timeout}_\text{retrans})\};
\ast \text{NextSeq}_{\text{seq}_\text{seq}}(\text{sn}_\text{seq}), & \text{ if } (u, v) = (\text{receive}_\text{mess}, \text{recv}_\text{seq}_\text{no});
\ast \text{NextSeq}_{\text{seq}_\text{seq}}(\text{sn}_\text{seq}), & \text{ if } (u, v) = (\text{receive}_\text{ack}, \text{send}_\text{seq}_\text{no});
\ast \text{IncrementRSN}_{\text{seq}_\text{seq}}(\text{sn}_\text{seq}, \text{rn}_\text{seq}), & \text{ if } (u, v) = (\text{receive}_\text{mess}, \text{recv}_\text{seq}_\text{no});
\ast \text{IncrementRSN}_{\text{seq}_\text{seq}}(\text{sn}_\text{seq}, \text{rn}_\text{seq}), & \text{ if } (u, v) = (\text{send}_\text{ack}, \text{ack}_\text{channel});
\ast \text{Loss}_{\text{messlist}_\text{seq}_\text{seq}}(\text{sn}_\text{seq}), & \text{ if } (u, v) = (\text{mess}_\text{loss}, \text{mess}_\text{channel});
\ast \text{Loss}_{\text{messlist}_\text{seq}_\text{seq}}(\text{sn}_\text{seq}), & \text{ if } (u, v) = (\text{ack}_\text{loss}, \text{ack}_\text{channel}).
\end{align*}
\]

\[
\begin{align*}
\text{TCond}(t) = \left\{ \begin{array}{ll}
\text{RetransEnabled}_{\text{retranscounter}, \text{Bool}}(\text{rc}_{\text{retranscounter}}), & \text{if } t = \text{timeout}_\text{retrans}, \\
\text{=(seq,Bool)}(\text{rn}_\text{seq}, \text{NextSeq}_{\text{seq}_\text{seq}}(\text{sn}_\text{seq})), & \text{if } t = \text{receive}_\text{ack}, \\
\text{<>(seq,Bool)}(\text{rn}_\text{seq}, \text{NextSeq}_{\text{seq}_\text{seq}}(\text{sn}_\text{seq})), & \text{if } t = \text{receive}_\text{dup}_\text{ack}, \\
\text{Contains}_{(\text{messlist}_\text{seq}_\text{seq}, \text{Bool})}(\text{queue}_\text{messlist}, \text{sn}_\text{seq}), & \text{if } t = \text{mess}_\text{loss}, \\
\text{Contains}_{(\text{messlist}_\text{seq}_\text{seq}, \text{Bool})}(\text{queue}_\text{messlist}, \text{rn}_\text{seq}), & \text{if } t = \text{ack}_\text{loss}, \\
\text{true}_\text{Bool} & \text{otherwise.}
\end{array} \right.
\end{align*}
\]
5.7. Properties for Verification

\[ M_0(p) = \begin{cases} 
\{(s\text{ready}, 1)\}, & \text{if } p = \text{sender\_state}, \\
\{(r\text{ready}, 1)\}, & \text{if } p = \text{receiver\_state}, \\
\{(0, 1)\}, & \text{if } p \in \{\text{send\_seq\_no, recv\_seq\_no, retrans\_counter}\}, \\
\{([], 1)\}, & \text{if } p \in \{\text{mess\_channel, ack\_channel}\}, \text{ where } [] \text{ represents the empty list.}
\end{cases} \]

In \( HLPNG_{(MS;MR)} \) we have made three notable changes from \( CPN_{(MS;MR)} \):

1. The guard of \( \text{timeout\_retrans} \), \( rc < \text{MaxRetrans} \), has been replaced by the operator \( \text{RetransEnabled}(\text{retranscounter,Bool}) \), to hide \( \text{MaxRetrans} \);

2. The inscription on the arc from \( \text{timeout\_retrans} \) to \( \text{retrans\_counter} \), \( rc + 1 \), has been replaced by the operator \( \text{IncrementRC}(\text{retranscounter, retranscounter}) \), to hide \( \text{MaxRetrans} \); and

3. The inscription on the arc from \( \text{receive\_mess} \) to \( \text{recv\_seq\_no} \), if \( (sn = rn) \) then \( \text{NextSeq}(rn) \) else \( rn \), has been replaced by the operator \( \text{IncrementRSN}(\text{retranscounter, retranscounter, retranscounter}) \), to cope with the conditional statement.

All three changes have been made to simplify the HLPNG representation by hiding details of arc inscriptions inside functions. Note that the bodies of the functions corresponding to these three new operators are identical to the corresponding inscriptions in \( CPN_{(MS;MR)} \), hence the model will behave in exactly the same way as \( CPN_{(MS;MR)} \). This is no different to using the \( \text{NextSeq}, \text{Contains} \) and \( \text{Loss} \) functions in \( CPN_{(MS;MR)} \), where the details of these functions are removed from the arcs and guards in Fig. 5.5 and placed in the declarations in Fig. 5.6 instead.

Note that in the CPN, we use infix notation for some of the functions, e.g. the ‘not equals to’ operator in the guard of the \( \text{receive\_dup\_ack} \) transition, \( [rc <> \text{NextSeq}(sn)] \). In a HLPNG, normal function notation is used, where the function name is followed by the arguments. For example, this same guard is expressed as \( <>_{(seq, seq, Bool)}(rn_{seq}, sn_{seq}) \).

5.7 Properties for Verification

The Stop-and-Wait Protocol must satisfy a number of properties in order to operate correctly. Our goal is to verify conformance to these properties parametrically, i.e. for all values of the parameters, in Chapters 9 and 10. We define these properties below.

5.7.1 Absence of Unexpected Dead Markings

Consider a situation in which the sender is waiting for an acknowledgement, has attempted to retransmit the maximum number of times, and both channels are empty (marked with the empty list). This is a
deadlock, corresponding to the sender giving up retransmission, as discussed in Section 5.1. This may happen for two reasons: all instances of the currently outstanding message were lost because of a broken or extremely unreliable message channel, in which case the receiver will still be expecting this message; or through any combination of message and acknowledgement loss, in which case the receiver will be expecting the next message if at least one instance of the currently outstanding message was received correctly. In either case, the resultant dead marking is expected (recall that the management entity that deals with this situation is not modelled) and hence is not a deadlock. All other dead markings are considered undesirable, and hence are deadlocks. We can formally define this property as follows.

**Property 5.1** (Absence of Unexpected Dead Markings). For each combination of MaxSeqNo and MaxRetrans, the Stop-and-Wait protocol must contain no deadlocks, i.e.

\[
\forall M \in [M_0] \setminus (V_{term1} \cup V_{term2}), \exists t \in T \text{ such that } M[t]
\]

where \(M[t]\) denotes that \(t\) is enabled in \(M\) in some mode, and \(V_{term1}\) and \(V_{term2}\) are given by

\[
V_{term1} = \{M \in [M_0] \mid M(sender\_state) = 1'wait\_ack, \\
M(receiver\_state) = 1'\text{ready}, \\
M(send\_seq\_no) = M(recv\_seq\_no), \\
M(retrans\_counter) = 1'MaxRetrans, \\
M(mess\_channel) = M(ack\_channel) = []\}\n\]

\[
V_{term2} = \{M \in [M_0] \mid M(sender\_state) = 1'wait\_ack, \\
M(receiver\_state) = 1'\text{ready}, \\
M(send\_seq\_no) \oplus_{MS} 1 = M(recv\_seq\_no), \\
M(retrans\_counter) = 1'MaxRetrans, \\
M(mess\_channel) = M(ack\_channel) = []\}\n\]

where \(\oplus_{MS}\) is modulo MaxSeqNo + 1 addition.

5.7.2 Absence of Livelock

Absence of livelock means that the SWP should not enter cycles in which no progress is being made, and from which the protocol cannot escape [17]. Section 3.1.3 suggests a way that livelocks can be detected using Strongly Connected Components [81]. This thesis does not consider SCCs, however a sufficient condition for absence of livelock is that all markings can reach a dead marking. This means that loops in the reachability graph with no arcs exiting the loop (a livelock) cannot exist. In Chapter 9 we discover that this sufficient condition is amenable to the analysis of our system. Formally:
5.7. Properties for Verification

Property 5.2 (Absence of Livelock). The Stop-and-Wait Protocol must contain no livelocks. A sufficient condition for the absence of livelock is that all markings can reach a dead marking, i.e.

\[ \forall M \in [M_0], \exists M' \in [M] \text{ such that } M' \text{ is a dead marking, i.e. } \exists t \in T \text{ such that } M'[t] \]

5.7.3 Absence of Unexpected Dead Transitions

Dead transitions indicate procedures that are specified by the protocol but which are never used. Dead procedures should be eliminated from the protocol unless there is an overriding reason for their presence, e.g. facilitating future expansion of the model. When \( \text{MaxRetrans}=0 \), the \( \text{timeout/retrans} \) transition will never be enabled and thus should be dead. Because there are no retransmissions, duplicate messages (and thus duplicate acknowledgements) should never be present in the system, and so it would be expected that \( \text{receive/dup/ack} \) should also be dead when \( \text{MaxRetrans}=0 \). Formally:

Property 5.3 (Absence of Unexpected Dead Transitions). The Stop-and-Wait Protocol CPN model should contain no unexpected dead transitions, i.e.

For \( \text{MaxRetrans} = 0 \),

\[ \forall t \in T \setminus \{\text{timeout/retrans, receive/dup/ack}\}, \exists M \in [M_0] \text{ such that } M[t], \text{ and} \]

\[ \forall t \in \{\text{timeout/retrans, receive/dup/ack}\}, \forall M \in [M_0], \exists t \in T \text{ such that } M[t] \]

For \( \text{MaxRetrans} > 0 \), \( \forall t \in T, \exists M \in [M_0] \text{ such that } M[t] \)

This property assumes a lossy medium. If the medium were lossless, the property would exclude the \( \text{mess/loss} \) and \( \text{ack/loss} \) transitions, as they can never be enabled.

5.7.4 Channel Bounds

A bound on the number of messages and acknowledgements in the channel is necessary to ensure that the protocol will not cause congestion collapse in the underlying network. Conversely, if the number of messages and acknowledgements is bounded, and if this bound is known, then the network can be designed with sufficient capacity. Formally:

Property 5.4 (Channel Bounds). The number of messages and acknowledgements in the underlying medium is bounded, i.e. the number of messages is bounded by \( n \in \mathbb{N} \) iff

\[ \forall M \in [M_0], |f_c(M(\text{mess/channel}))| \leq n \]

and the number of acknowledgements is bounded by \( n \in \mathbb{N} \) iff

\[ \forall M \in [M_0], |f_c(M(\text{ack/channel}))| \leq n \]

where \( |x| \) denotes the length of list \( x \).
5.8. Concluding Remarks

This property relies on all reachable markings having exactly one token in this place. This is proved in Section 6.3.

5.7.5 Conformance to the Service Language

This property is discussed in Section 5.2.

Property 5.5 (Conformance of the SWP to its Service Language). The Stop-and-Wait Protocol must conform to the Stop-and-Wait Property of alternating Send and Receive events given by Definition 5.1 and illustrated in Fig. 5.3.

5.8 Concluding Remarks

This chapter has introduced stop-and-wait flow control, described and modelled the Stop-and-Wait Protocol, defined a service of alternating send and receive events, and defined a set of properties for verification. The next chapter describes some preliminary investigations of the SWP CPN model presented in this section, firstly without retransmissions (MaxRetrans=0) and then in both MaxRetrans and MaxSeqNo.
Chapter 6

Model Behaviour

The first step undertaken toward formulating a parametric reachability graph for the Stop-and-Wait CPN model was the discovery of a regular structure in the reachability graphs for different parameter values. Section 6.1 presents an examination of the concrete reachability graphs of the SWP CPN for the base case where MaxRetrans=0, for small values of MaxSeqNo, the discovery of repeated patterns of behaviour, and algebraic expressions for the parametric reachability graph in the MaxSeqNo parameter (MaxRetrans=0) from [48, 49]. In Section 6.2 we examine reachability graphs of the SWP CPN model when retransmissions are considered. Evidence of similar (but more complex) repeated patterns of behaviour is uncovered, and a conjecture for the size of the reachability graph in terms of the two parameters is stated. Finally, Section 6.3 formalises important behavioural properties used when defining symbolic notation for markings and arcs (Chapter 7), in the proof of correctness of the parametric reachability graph, and in its analysis (Chapters 8 to 10).

Before we proceed further, let us introduce notation for the parameterised reachability graph of $CPN_{(MS,MR)}$: From Definition 2.14, the reachability graph of $CPN_{(MS,MR)}$ is explicitly denoted by the parameterised reachability graph $RG_{(MS,MR)} = (V_{(MS,MR)}, A_{(MS,MR)})$.

6.1 The SWP Without Retransmissions

Using Design/CPN [39] the reachability graph of $CPN_{(MS,0)}$ (i.e. no retransmissions) was generated for $1 \leq MS \leq 4$ using Design/CPN’s Occurrence Graph tool [31]. $RG_{(1,0)}$ is shown in Fig. 6.1(a) and consists of 12 nodes and 12 arcs. It has a main loop of 8 nodes with four dead markings branching from it. The nodes are numbered from 1 to 12 and each node has been annotated with its sender and receiver sequence number, written as the pair $(f_c(M(send,seq,no)), f_c(M(recv,seq,no)))$ for reasons that will become evident. For example, node 4 is annotated with (0,1) indicating that the sender sequence number is 0 and the receiver sequence number is 1. Node 4 represents the marking in which:
6.1. The SWP Without Retransmissions

![Diagram of RGs]

(a) $RG_{(1,0)}$

(b) $RG_{(2,0)}$

Figure 6.1: The RGs of $CPN_{(1,0)}$ and $CPN_{(2,0)}$: a) $RG_{(2,0)}$; b) $RG_{(2,0)}$.

\[
\begin{align*}
M(\text{sender\_state}) & = 1\cdot \text{wait\_ack} & M(\text{send\_seq\_no}) & = 1\cdot 0 \\
M(\text{receiver\_state}) & = 1\cdot \text{process} & M(\text{recv\_seq\_no}) & = 1\cdot 1 \\
M(\text{mess\_channel}) & = 1\cdot \text{[]} & M(\text{ack\_channel}) & = 1\cdot \text{[]} \\
M(\text{retrans\_counter}) & = 1\cdot 0 \\
\end{align*}
\]

Each arc is labelled with the corresponding action from the CPN, including the sequence number of the message or acknowledgement where necessary for clarity. For example, the label $\text{send\_mess \_0}$ on the arc from node 1 to node 2 represents the occurrence of transition $\text{send\_mess}$ with $\text{sn}$ bound to 0 (the sending of a message with sequence number 0).
6.1. The SWP Without Retransmissions

The main loop represents the behaviour of the Stop-and-Wait protocol when no messages are lost. As there are no retransmissions, the dead markings arise when either a message or an acknowledgement is lost. The message or the acknowledgement may be lost for each sequence number, so we have a total of four dead markings.

\[ \text{RG}(2,0) \] given in Fig 6.1(b) contains 18 nodes and 18 arcs, with 6 dead markings. A main loop similar to that in \( \text{RG}(1,0) \) is present, but consisting of 12 nodes instead of 8 because \( MS = 2 \), thus the message with sequence number 2 must be sent and acknowledged before sequence numbers wrap and the protocol can return to the initial state. This new message and acknowledgement may be lost, resulting in the two additional dead markings.

6.1.1 Regular Patterns in the Concrete Reachability Graphs

From [48, 49], there is a regular structure to the RGs when \( MR = 0 \). The annotation of the nodes with the sender/receiver sequence number pair allows us to see that for each message sent (i.e. for each sequence number) there are 6 markings generated that can be partitioned into two sets of 3 markings each. The sender and receiver sequence numbers are identical for the first set of three markings which comprise the markings where: 1. the acknowledgement has been received by the sender (or it is the initial marking); 2. the message has been sent; and 3. the message has been lost. The set of nodes, \{1, 2, 3\} (for sender sequence number 0), is an example of this first set of 3 markings. The second set of 3 markings corresponds to the receive sequence number being one more than the send sequence number (modulo \( MS + 1 \)) and comprises the markings where: 1. the message has been received; 2. the acknowledgement has been sent; and 3. the acknowledgement has been lost. The set of nodes, \{4, 5, 6\} (for sender sequence number 0), is an example of this second set of 3 markings.

The subgraphs associated with the subsets of 6 nodes that each have the same sender sequence number (or subgraphs associated with the subsets of 6 nodes that each have the same receiver sequence number, although we do not consider this) all exhibit this same behaviour. This is true both across the subgraphs within each RG in Fig. 6.1 and the subgraphs across the two RGs in Fig. 6.1. This repeating pattern of behaviour continues as \( MS \) increases and is evident in \( \text{RG}(3,0) \) and \( \text{RG}(4,0) \) included in Appendix B.

Based on these observations, the regular behaviour can be formalised as follows. Let \( V^{(MS,0)}_{(ssn,rsn)} \subseteq V^{(MS,0)} \) denote the subset of nodes of \( \text{RG}^{(MS,0)} \) with sender sequence number equal to \( ssn \) and receiver sequence number equal to \( rsn \), i.e. \( ssn = f_c(M(\text{send.seq.no})) \) and \( rsn = f_c(M(\text{recv.seq.no})) \). For each message sent with sequence number \( i \in \{0, ..., MS\} \), the first set of three nodes described above correspond to those in which \( ssn = rsn \) and is denoted \( V^{(MS,0)}_{(i,i)} \) (see Table 6.1). The second set of three nodes described above correspond to those in which \( rsn = ssn \oplus MS \) 1 (where \( \oplus MS \) is modulo
6.1. The SWP Without Retransmissions

Table 6.1: $V^{(MS,0)}_{(i,i)}$ for $0 \leq i \leq MS$

<table>
<thead>
<tr>
<th>Node</th>
<th>sender_state</th>
<th>sendSeqNo</th>
<th>mess_ch.</th>
<th>ack_ch.</th>
<th>receiver_state</th>
<th>recvSeqNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M^{(MS,0)}_{(1,i,i)}$</td>
<td>1’s_ready</td>
<td>1’i</td>
<td>1’[]</td>
<td>1’[]</td>
<td>1’r_ready</td>
<td>1’i</td>
</tr>
<tr>
<td>$M^{(MS,0)}_{(2,i,i)}$</td>
<td>1’wait_ack</td>
<td>1’i</td>
<td>1’[i]</td>
<td>1’[]</td>
<td>1’r_ready</td>
<td>1’i</td>
</tr>
<tr>
<td>$M^{(MS,0)}_{(3,i,i)}$</td>
<td>1’wait_ack</td>
<td>1’i</td>
<td>1’[]</td>
<td>1’[]</td>
<td>1’r_ready</td>
<td>1’i</td>
</tr>
</tbody>
</table>

Table 6.2: $V^{(MS,0)}_{(i,i;MS1)}$ for $0 \leq i \leq MS$

<table>
<thead>
<tr>
<th>Node</th>
<th>sender_state</th>
<th>sendSeqNo</th>
<th>mess_ch.</th>
<th>ack_ch.</th>
<th>receiver_state</th>
<th>recvSeqNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M^{(MS,0)}_{(1,i,i;MS1)}$</td>
<td>1’wait_ack</td>
<td>1’i</td>
<td>1’[]</td>
<td>1’[]</td>
<td>1’process</td>
<td>1’i⊕MS 1</td>
</tr>
<tr>
<td>$M^{(MS,0)}_{(2,i,i;MS1)}$</td>
<td>1’wait_ack</td>
<td>1’i</td>
<td>1’[i]</td>
<td>1’[i⊕MS 1]</td>
<td>1’r_ready</td>
<td>1’i⊕MS 1</td>
</tr>
<tr>
<td>$M^{(MS,0)}_{(3,i,i;MS1)}$</td>
<td>1’wait_ack</td>
<td>1’i</td>
<td>1’[]</td>
<td>1’[]</td>
<td>1’r_ready</td>
<td>1’i⊕MS 1</td>
</tr>
</tbody>
</table>

(MS + 1) addition) and is denoted $V^{(MS,0)}_{(i,i;MS1)}$ (see Table 6.2). Thus, for $0 \leq i \leq MS$:

$$V^{(MS,0)}_{(i,i)} = \{ M \in V^{(MS,0)} | f_c(M(\text{recv_seq_no})) = f_c(M(\text{send_seq_no})) = i \}$$

and

$$V^{(MS,0)}_{(i,i;MS1)} = \{ M \in V^{(MS,0)} | f_c(M(\text{send_seq_no})) = f_c(M(\text{recv_seq_no})) = i \oplus MS 1 \}$$

The markings are given in Tables 6.1 and 6.2 respectively. The first column shows the marking name. The superscript of a marking corresponds to the two parameters, $MS$ and $MR$, where $MR = 0$. The subscript of a marking, $(\text{class}, \text{ssn}, \text{rsn})$, is the class of marking (either 1, 2 or 3) as described above, and the sender and receiver sequence number for that marking. The remaining columns of these tables show the markings of each place in the SWP CPN of Fig. 5.5. Note that because $MR = 0$ the timeout_retrans transition is never enabled, and so retrans_counter always has a value of 0 (one ‘0’ token) and thus is omitted from the tables.

To match the partitioning of nodes, we define a similar partitioning of the arcs into two groups of three. Let $A^{(MS,0)}_{(ssn,rsn)} \subset A^{(MS,0)}$ denote the subset of arcs in $RG^{(MS,0)}$ in which the source node of the arc has the sender sequence number $ssn$ and receiver sequence number $rsn$. As before, $rsn = ssn$ or $ssn \oplus MS 1$. For $0 \leq i \leq MS$ we define:

$$A^{(MS,0)}_{(i,i)} = \{ (M, (t, b), M') \in A^{(MS,0)} | f_c(M(\text{recv_seq_no})) = f_c(M(\text{send_seq_no})) = i \}$$
6.1. The SWP Without Retransmissions

Table 6.3: $A^{(MS,0)}_{(i,i)}$ for $0 \leq i \leq MS$

<table>
<thead>
<tr>
<th>Name</th>
<th>Source node</th>
<th>Binding Element</th>
<th>Dest. Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^{(MS,0)}_{(1,i,i)}$</td>
<td>$M^{(MS,0)}_{(1,i,i)}$</td>
<td>(send$_{-}$mess, $(sn = i, queue = [])$)</td>
<td>$M^{(MS,0)}_{(2,i,i)}$</td>
</tr>
<tr>
<td>$a^{(MS,0)}_{(2,i,i)}$</td>
<td>$M^{(MS,0)}_{(2,i,i)}$</td>
<td>(mess$_{-}$loss, $(sn = i, queue = [])$)</td>
<td>$M^{(MS,0)}_{(3,i,i)}$</td>
</tr>
<tr>
<td>$a^{(MS,0)}_{(3,i,i)}$</td>
<td>$M^{(MS,0)}_{(2,i,i)}$</td>
<td>(receive$_{-}$mess, $(sn = i, rn = i, queue = [])$)</td>
<td>$M^{(MS,0)}_{(1,i,i \oplus MS1)}$</td>
</tr>
</tbody>
</table>

Table 6.4: $A^{(MS,0)}_{(i,i \oplus MS1)}$ for $0 \leq i \leq MS$

<table>
<thead>
<tr>
<th>Name</th>
<th>Source node</th>
<th>Binding Element</th>
<th>Dest. Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^{(MS,0)}_{(1,i,i \oplus MS1)}$</td>
<td>$M^{(MS,0)}_{(1,i,i \oplus MS1)}$</td>
<td>(send$_{-}$ack, $(rn = i \oplus MS 1, queue = [])$)</td>
<td>$M^{(MS,0)}_{(2,i,i \oplus MS1)}$</td>
</tr>
<tr>
<td>$a^{(MS,0)}_{(2,i,i \oplus MS1)}$</td>
<td>$M^{(MS,0)}_{(2,i,i \oplus MS1)}$</td>
<td>(ack$_{-}$loss, $(rn = i \oplus MS 1, queue = [])$)</td>
<td>$M^{(MS,0)}_{(3,i,i \oplus MS1)}$</td>
</tr>
<tr>
<td>$a^{(MS,0)}_{(3,i,i \oplus MS1)}$</td>
<td>$M^{(MS,0)}_{(2,i,i \oplus MS1)}$</td>
<td>(receive$_{-}$ack, $(sn = i, rn = i \oplus MS 1, rc = 0, queue = [])$)</td>
<td>$M^{(MS,0)}_{(1,i,i \oplus MS1 \oplus MS1)}$</td>
</tr>
</tbody>
</table>

and

$$A^{(MS,0)}_{(i,i \oplus MS1)} = \{(M, (t, b), M') \in A^{(MS,0)} | f_t(M(\text{send}\_\text{seq}\_\text{no})) = i,$$

$$f_t(M(\text{recv}\_\text{seq}\_\text{no})) = i \oplus MS 1\}$$

where $(t, b)$ and $M'$ are given by Tables 6.3 and 6.4 (described shortly).

Partitioning the arcs in this way means that $A^{(MS,0)}_{(sn,rn)}$ contains all the outgoing arcs of the nodes in the corresponding set, $V^{(MS,0)}_{(sn,rn)}$, for a given $sn$ and $rn$. The arcs in $A^{(MS,0)}_{(i,i)}$ represent: 1. sending a message with sequence number $i$; 2. losing this message; and 3. receiving this message. The arcs in $A^{(MS,0)}_{(i,i \oplus MS1)}$ represent: 1. sending an acknowledgement, $i \oplus MS 1$; 2. loss of this acknowledgement; and 3. receiving this acknowledgement. These arcs are given in Tables 6.3 and 6.4. The first column gives the name of the arc. Again, the superscript corresponds to the two parameters, $MS$ and $MR$, where $MR = 0$. The subscript indicates the class, sender sequence number and receiver sequence number of the source node of each arc. The second column gives the source marking. The binding element is given in the third column, and the destination marking is given in the fourth column.

Tables 6.1 to 6.4 describe a repeating pattern of 6 nodes and 6 arcs. Based on a diagram from [48,49], we can visualise this repeating pattern as shown in Fig. 6.2. In this figure, nodes have been labelled by their parametric name, whereas arcs have been labelled by the corresponding transition, rather than their parametric arc name or binding element. This is for simplicity and to give the reader more of an immediate feel for the structure and repeating behaviour of the reachability graph. The last node and arc,
6.1. The SWP Without Retransmissions

Figure 6.2: The fragment of $RG_{(MS,0)}$ for any $i$, $0 \leq i \leq MS$.

at the bottom of the figure, are dashed to indicate that they are part of the next repetition of this pattern of 6 nodes and 6 arcs.

6.1.2 Parametric Reachability Graph in MaxSeqNo

From Tables 6.1 to 6.4, an expression for the parametric reachability graph when $MR = 0$, $RG_{(MS,0)}$, can be derived. We know that the set of nodes of $RG_{(1,0)}$ is exactly specified by the union of the evaluation of the sets $V_{(i,i)}^{(MS,0)}$ and $V_{(i,i\oplus MS)}^{(MS,0)}$ for each value of $i$ from 0 to $MS$, which can be confirmed by direct inspection of $RG_{(1,0)}$ in Fig. 6.1(a). Similarly, the set of arcs of $RG_{(1,0)}$ is exactly specified by the union of the evaluation of the sets $A_{(i,i)}^{(MS,0)}$ and $A_{(i,i\oplus MS)}^{(MS,0)}$ for all values of $i$ from 0 to $MS$. The same is true of $RG_{(2,0)}$, and so on. We are therefore able to specify the RG directly, for any value of $MS \in \mathbb{N}^+$, as a union of sets of nodes and a union of sets of arcs. We state the theorem (from [48, 49]) for this parametric RG below.

Theorem 6.1. For $MS \in \mathbb{N}^+$, $RG_{(MS,0)} = (V_{(MS,0)}, A_{(MS,0)})$ where

$$V_{(MS,0)} = \bigcup_{0 \leq i \leq MS} V_{(i,i)}^{(MS,0)} \cup V_{(i,i\oplus MS)}^{(MS,0)}, \quad A_{(MS,0)} = \bigcup_{0 \leq i \leq MS} A_{(i,i)}^{(MS,0)} \cup A_{(i,i\oplus MS)}^{(MS,0)}$$
The proof of this theorem is given in [48,49] and is covered by the parametric reachability graph in both parameters (Chapter 8) and so is not reproduced here.

## 6.2 The SWP With Retransmissions

We conjecture that this repeating pattern of behaviour continues in the presence of retransmissions, however the reachability graphs become larger much more quickly and are too complex to easily visualise. (For example, with $\text{MaxSeqNo}$ fixed to 1, as the number of retransmissions grows from 0 to 1 to 2, the number of nodes grows from 12 to 92 to 336, and the number of arcs grows from 12 to 242 to 1166.)

Reachability graphs were generated for instantiations of $CPN_{(MS,MR)}$ for the range of parameter values $1 \leq MS \leq 11, 0 \leq MR \leq 10$. Selected statistics are shown in Table 6.5, and the complete table is given in Appendix C (Table C.1). The first two columns show the values of the $\text{MaxRetrans}$ and $\text{MaxSeqNo}$ parameters. The next two columns show the number of nodes and arcs in each reachability graph. The last column shows the number of dead markings present in each RG. Similar statistics have also been presented in [16, 17, 48] but using the less general model of loss, hence the statistics are different.

For a given $\text{MaxRetrans}$ the number of nodes and arcs increases linearly with $\text{MaxSeqNo}$. For example, when $\text{MaxRetrans} = 0$ the number of nodes and arcs increases by 6 for each unit increase in $\text{MaxSeqNo}$. This corroborates the evidence of a repeating pattern identified in Section 6.1. When $\text{MaxRetrans} = 1$, the increase is 46 nodes and 121 arcs for each unit increase in $\text{MaxSeqNo}$. Applying standard techniques for fitting polynomials to data to Table C.1 in Appendix C (see Appendix C for details of these calculations) we conjecture that the size of $RG_{(MS,MR)}$ is given by:

**Conjecture 6.1.** For the Stop-and-Wait CPN of Figs. 5.5 and 5.6, the number of nodes in the reachability graph is given by:

$$|V_{(MS,MR)}| = ((MS + 1)/6)(5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36)$$

and the number of arcs is given by

$$|A_{(MS,MR)}| = ((MS + 1)/6)(30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)$$

The range of parameter values investigated gives us a high degree of confidence in this result.

Before developing algebraic expressions for the parametric RG, however, this conjecture is useful for gaining an insight into the structure of the RG in both $\text{MaxSeqNo}$ and $\text{MaxRetrans}$ parameters. Firstly, the size of $RG_{(MS,MR)}$ is factorable into the two parameters. This gives a strong indication that there is also repeating behaviour for each sender sequence number in $RG_{(MS,MR)}$ for $MR > 0$, not just...
6.2. The SWP With Retransmissions

Table 6.5: RG Statistics of the CPN in Figs. 5.5 and 5.6 for various parameter values.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>MaxSeqNo</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Dead Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>12</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>18</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>24</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>30</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>60</td>
<td>60</td>
<td>20</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>66</td>
<td>66</td>
<td>22</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>92</td>
<td>242</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>138</td>
<td>363</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>184</td>
<td>484</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>230</td>
<td>605</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>460</td>
<td>1210</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>506</td>
<td>1331</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>336</td>
<td>1166</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>504</td>
<td>1749</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>672</td>
<td>2332</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>840</td>
<td>2915</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>1680</td>
<td>5830</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1848</td>
<td>6413</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>880</td>
<td>3494</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1320</td>
<td>5241</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1760</td>
<td>6988</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2200</td>
<td>8735</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>4400</td>
<td>17470</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>4840</td>
<td>19217</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1900</td>
<td>8176</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2850</td>
<td>12264</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3800</td>
<td>16352</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4750</td>
<td>20440</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>9500</td>
<td>40880</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>10450</td>
<td>44968</td>
<td>22</td>
</tr>
</tbody>
</table>
$\text{MR} = 0$. Secondly, the size expression is linear in $\text{MS}$ and quartic in $\text{MR}$. An increase in the value of the $\text{MaxRetrans}$ parameter causes a quartic increase in the number of nodes and arcs, regardless of the value of $\text{MS}$. No longer are we dealing with a simple repeating pattern of 6 nodes and 6 arcs when $\text{MR} = 0$!

6.3 Investigating Behavioural Properties

A number of behavioural properties of the CPN from Chapter 5 are formalised in this section. These properties facilitate the definition of notation for the parametric RG in Chapter 7 and are used in the proof of correctness of the parametric RG in Chapter 8.

6.3.1 Singleton Multiset Markings

We can prove from the CPN structure that every marking of every place is a singleton multiset, in all reachable markings of the CPN. This is needed when defining the parametric notation for markings and arcs:

**Lemma 6.1.** For all reachable markings of $\text{CPN}_{(\text{MS,MR})}$, each place in the CPN diagram contains exactly one token, i.e. $\forall M \in \text{V}_{(\text{MS,MR})}, |M(\text{sender state})| = |M(\text{receiver state})| = |M(\text{send seq no})| = |M(\text{recv seq no})| = |M(\text{mess channel})| = |M(\text{ack channel})| = |M(\text{retrans counter})| = 1$.

**Proof.** Proof is by direct inspection of the CPN in Figs. 5.5 and 5.6. Every transition in the CPN diagram in Fig. 5.5 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 $\text{MaxSeqNo}$ parameter, through function $\text{NextSend}$ defined in the declarations in Fig. 5.6) but the parameter values do not affect the number of tokens removed or produced. We systematically examine each of the places below.

**sender state** $|M_0(\text{sender state})| = |1^{s\text{,ready}}| = 1$, hence this lemma holds for the initial marking. The marking of this place can only be changed by the $\text{send mess}$, $\text{timeout retrans}$ and $\text{receive ack}$ transitions. The occurrence of these transitions either replaces one value by another (the $\text{send mess}$ and $\text{receive ack}$ transitions) or does not change the marking (the $\text{timeout retrans}$ transition). Hence $\forall M \in [M_0], |M(\text{sender state})| = 1$.

**receiver state** $|M_0(\text{receiver state})| = |1^{r\text{,ready}}| = 1$, hence this lemma holds for the initial marking. The marking of this place can only be changed by the $\text{receive mess}$ and $\text{send ack}$
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 \text{send.mess}, \text{timeout.retrans}, \text{receive.ack} and \text{receive.dup.ack} transitions. The occurrence of these transitions either replaces one value by another (the \text{receive.ack} transition) or does not change the marking (the \text{send.mess}, \text{timeout.retrans} and \text{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 \text{recv.mess} and \text{send.ack} transitions. The occurrence of these transitions either replaces one value by another (the \text{recv.mess} transition in the case of receiving a new message) or does not change the marking (the \text{recv.mess} transition in the case of receiving a duplicate message and the \text{send.ack} transition). Hence \( \forall M \in [M_0], |M(\text{recv_seq_no})| = 1. \)

**mess_channel** \( |M_0(\text{mess_channel})| = |1'||| = 1 \), hence this lemma holds for the initial marking. The marking of this place can only be changed by the \text{send.mess}, \text{timeout.retrans}, \text{mess.loss} and \text{recv.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** \( |M_0(\text{ack_channel})| = |1'||| = 1 \), hence this lemma holds for the initial marking. The marking of this place can only be changed by the \text{send.ack}, \text{ack.loss}, \text{receive.ack} and \text{recv.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 \text{timeout.retrans} and \text{recv.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.

**Corollary 6.1.** The function, \( f_c \), converting a singleton multiset to a value, can be applied to the marking of any place in any reachable marking of \( CPN_{(MS, MR)} \).
6.3. Investigating Behavioural Properties

6.3.2 Sequence Number Independent Behaviour

In Section 6.1 we identified and formalised a pattern of behaviour that repeats for every sender sequence number between 0 and \( MS \) when \( MR = 0 \). In Section 6.2 we discovered evidence that a similar (but more complex) pattern of behaviour may exist when \( MR > 0 \). Intuition, therefore, suggests that the behaviour of each transition in \( CPN_{(MS,MR)} \) may be in some way independent of the sequence numbers over which it is operating. For example, the \( \text{send mess} \) transition will be enabled when an \( s\text{ready} \) token is present in the \( \text{sender state} \) place, and the \( \text{send seq no} \) and \( \text{mess channel} \) places contain at least one token each, regardless of the actual values of the sequence numbers involved.

We use the enabling and transition rules of a HLPNG given in Definition 2.11 and 2.12 respectively, and \( HLPNG_{(MS,MR)} \) from Section 5.6, to prove a series of lemmas, one for each transition, defining the symbolic enabling and occurrence rules for each transition. This captures the sequence number independent behaviour of each transition in our parameterised SWP model.\(^1\) The first part of each lemma specifies a set of reachable markings in terms of the characteristics that each marking must have in order to enable the corresponding transition, which shows that the form of the marking which will enable a transition is independent of the concrete sequence numbers involved. The second part of each lemma specifies the form of the markings resulting from the occurrence of the corresponding transition, thus demonstrating that the form of the resulting markings are the same, regardless of the concrete sequence numbers involved. The notation \(|x|\) is used to represent either the size (cardinality) of a multiset (as in Definition 2.5) if \( x \) is a multiset, or the length of a list, if \( x \) is a list value.

**Lemma 6.2.** For the \( \text{send mess} \) transition:

1. Given \( V = \{M \mid M \in [M_0] \wedge M(\text{sender state}) = 1\text{s_ready}, M \in V \implies M[\text{send mess}] \}. \)

2. Given \( V' = \{M' \mid M[\text{send mess}]M' \wedge M \in V \}, M' \in V' \implies \)
   - \( M'(\text{sender state}) = 1\text{wait ack}; \)
   - \( M'(\text{mess channel}) = 1\text{rc}(M(\text{mess channel}))^{\text{rc}}[\text{rc}(M(\text{send seq no}))]; \) and
   - \( \forall p \in P \setminus \{\text{sender state}, \text{mess channel}\}, M'(p) = M(p). \)

**Proof.** Part 1.

Recall from the enabling rule of HLPNGs (Definition 2.11):

A transition \( t \in T \) is enabled in a marking, \( M \), for a particular assignment, \( \alpha_t \), to its variables, that satisfies the transition condition, \( Val_{\text{Bool},\alpha}(TCond(t)) = \text{true} \), known as a mode of \( t \), iff

\[
\forall p \in P, Val_{\alpha}(\overline{p}, \overline{t}) \leq M(p)
\]

---

\(^1\)With thanks to J. Billington for his suggestion to use High-level Petri Net Graph notation for these lemmas, and for assistance in formulating the proof of Lemma 6.2.
6.3. Investigating Behavioural Properties

where for \((u,v) \in (P \times T) \cup (T \times P)\),

- \(\varpi, v = ATerm(u, v)\), for \((u,v) \in F\),

- \(\varpi, v = \Phi\), for \((u,v) \notin F\)

where \(\Phi\) is a symbol that represents the empty multiset at the level of the signature, so that \(Val_\alpha(\Phi) = \emptyset\).

From HLPNG\(_{(MS,MR)}\) and Definition 2.11, for a particular assignment, \(\alpha_{send\_mess}\), to its variables (i.e. for a particular binding of its variables):

1. \(send\_mess\) has input arcs from \(sender\_state\), \(send\_seq\_no\) and \(mess\_channel\), where:

   (a) \(ATerm\((sender\_state, send\_mess) = s\_ready\_sender\), and thus \(Val_{\alpha_{send\_mess}}(ATerm(sender\_state, send\_mess)) = 1's\_ready\).

   By Lemma 6.1, this condition is satisfied so long as \(M(sender\_state) = 1's\_ready\).

   (b) \(ATerm\(send\_seq\_no, send\_mess) = sn\_seq\), and thus \(Val_{\alpha_{send\_mess}}(ATerm(send\_seq\_no, send\_mess)) = 1'i\) for \(i \in Seq\).

   By Lemma 6.1, this condition is satisfied by all \(M \in [M_0]\), so long as \(\alpha_{send\_mess}(sn\_seq) = i\) when \(M(send\_seq\_no) = 1'i\).

   (c) \(ATerm\(mess\_channel, send\_mess) = queue\_messlist\), and thus \(Val_{\alpha_{send\_mess}}(ATerm(mess\_channel, send\_mess)) = 1'q\) for \(q \in MessList\).

   By Lemma 6.1, this condition is satisfied by all \(M \in [M_0]\), so long as \(\alpha_{send\_mess}(queue\_messlist) = q\) when \(M(mess\_channel) = 1'q\).

2. \(\forall p \in P \setminus \{sender\_state, send\_seq\_no, mess\_channel\}, \ Val_{\alpha_{send\_mess}}(p, send\_mess) = 0\), and \(M(p) \geq 0\) is true for all \(M \in [M_0]\).

3. \(TCond(send\_mess) = true\_Bool\), hence \(Val_{\alpha_{send\_mess}}(TCond(send\_mess)) = true\).

Hence \(send\_mess\) is enabled for the set of markings \(\{M \mid M \in [M_0] \land M(sender\_state) = 1's\_ready\}\).

Part 2.

Recall from the transition rule of HLPNGs (Definition 2.12):

If \(t \in T\) is enabled in mode \(\alpha_t\), for marking \(M\), \(t\) may occur in mode \(\alpha_t\). When \(t\) occurs in mode \(\alpha_t\), the marking of the net is transformed to a new marking \(M'\), denoted \(M[t, \alpha_t]M'\), according to the following rule:

\[
\forall p \in P, M'(p) = M(p) - Val_{\alpha_t}(p, t) + Val_{\alpha_t}(t, p)
\]

\(\text{By convention, we understand that there is an 'invisible' operator in the algebra of HLPNG\(_{(MS,MR)}\) that converts the result of } Val_{\alpha_t}, t \in T, \text{ from a value to its singleton multiset on evaluation.}\)
6.3. Investigating Behavioural Properties

Given $M$ enables $\text{send}_{\text{mess}}$, then from the first part of this proof we have $\alpha_{\text{send}_{\text{mess}}}(\text{sn}_{\text{seq}}) = i$ and $\alpha_{\text{send}_{\text{mess}}}(\text{queue}_{\text{messlist}}) = q$. From $\text{HLPNG}_{(MS,MR)}$, $\text{send}_{\text{mess}}$ has output arcs to sender_state, send_seq_no and mess_channel, where:

1. $\text{ATerm}(\text{send}_{\text{mess}}, \text{sender_state}) = \text{wait}_{\text{ack}_{\text{sender}}}$, and thus $\text{Val}_{\alpha_{\text{send}_{\text{mess}}}}(\text{ATerm}(\text{send}_{\text{mess}}, \text{sender_state})) = 1^i\text{wait}_{\text{ack}}$.

2. $\text{ATerm}(\text{send}_{\text{mess}}, \text{send_seq_no}) = \text{sn}_{\text{seq}}$, and thus $\text{Val}_{\alpha_{\text{send}_{\text{mess}}}}(\text{ATerm}(\text{send}_{\text{mess}}, \text{send_seq_no})) = 1^i$.

3. $\text{ATerm}(\text{send}_{\text{mess}}, \text{mess_channel}) = \text{MakeList}_{\text{seq, messlist}}(\text{sn}_{\text{seq}})$, and thus $\text{Val}_{\alpha_{\text{send}_{\text{mess}}}}(\text{ATerm}(\text{send}_{\text{mess}}, \text{mess_channel})) = 1^q$.$^i$.

4. $\forall p \in P \setminus \{\text{sender_state}, \text{send_seq_no}, \text{mess_channel}\}, (\text{send}_{\text{mess}}, p) \notin F$, hence $\text{Val}_{\alpha_{\text{send}_{\text{mess}}}}(\text{send}_{\text{mess}}, p) = \emptyset$.

Hence, from the transition rule:

1. $M'(\text{sender_state}) = M(\text{sender_state}) + 1^i\text{ready} + 1^i\text{wait}_{\text{ack}} = 1^i\text{wait}_{\text{ack}}$.

2. $M'(\text{send_seq_no}) = M(\text{send_seq_no}) + 1^i + 1^i = M(\text{send_seq_no}) = 1^i$.

3. $M'(\text{mess_channel}) = M(\text{mess_channel}) + 1^q + 1^q$.$^i[i] = 1^q$.$^i[i]$.

4. $\forall p \in P \setminus \{\text{sender_state}, \text{send_seq_no}, \text{mess_channel}\}, M'(p) = M(p)$.

Hence, $\forall M' \in V', M'(\text{sender_state}) = 1^i\text{wait}_{\text{ack}}, M'(\text{mess_channel}) = 1^i f_c(M(\text{mess_channel}))$.$^i$.$^i[f_c(M(\text{send_seq_no}))]$, and the marking of all other places is unchanged by the firing of $\text{send}_{\text{mess}}$.

Thus from the proofs of Part 1 and Part 2, the lemma is proved.

**Lemma 6.3.** For the $\text{timeout_retrans}$ transition:

1. Given $V = \{M \mid M \in [M_0] \land f_c(M(\text{retrans_counter})) < MR \land M(\text{sender_state}) = 1^i\text{wait}_{\text{ack}}\}, M \in V \implies M[\text{timeout_retrans}]$.

2. Given $V' = \{M' \mid M[\text{timeout_retrans}], M' \land M \in V\}, M' \in V' \implies$

   - $M'(\text{retrans_counter}) = 1^i(f_c(M(\text{retrans_counter})) + 1)$;
   - $M'(\text{mess_channel}) = 1^i f_c(M(\text{mess_channel}))$.$^i[f_c(M(\text{send_seq_no}))];$ and
   - $\forall p \in P \setminus \{\text{mess_channel}, \text{retrans_counter}\}, M'(p) = M(p)$.
6.3. Investigating Behavioural Properties

Using the same procedure as the proof of Lemma 6.2, from $HLPNG_{(MS,MR)}$ and Definition 2.11, for a particular assignment, $\alpha_{\text{timeout\_retrans}}$, to its variables:

1. $\text{timeout\_retrans}$ has input arcs from $\text{sender\_state}$, $\text{send\_seq\_no}$, $\text{mess\_channel}$ and $\text{retrans\_counter}$, where:

   (a) $ATerm(\text{sender\_state}, \text{timeout\_retrans}) = \text{wait\_ack\_sender}$, and thus
   \[
   Val_{\text{timeout\_retrans}}(ATerm(\text{sender\_state}, \text{timeout\_retrans})) = 1'\text{wait\_ack}.
   \]
   By Lemma 6.1, this condition is satisfied so long as $M(\text{sender\_state}) = 1'\text{wait\_ack}$.

   (b) $ATerm(\text{send\_seq\_no}, \text{timeout\_retrans}) = \text{sn\_seq}$, and thus
   \[
   Val_{\text{timeout\_retrans}}(ATerm(\text{send\_seq\_no}, \text{timeout\_retrans})) = 1'i \text{ for } i \in \text{Seq}.
   \]
   By Lemma 6.1, this condition is satisfied by all $M \in [M_0]$ so long as $\alpha_{\text{timeout\_retrans}}(\text{sn\_seq}) = i$ when $M(\text{send\_seq\_no}) = 1'i$.

   (c) $ATerm(\text{mess\_channel}, \text{timeout\_retrans}) = \text{queue\_messlist}$, and thus
   \[
   Val_{\text{timeout\_retrans}}(ATerm(\text{mess\_channel}, \text{timeout\_retrans})) = 1'q \text{ for } q \in \text{MessList}.
   \]
   By Lemma 6.1, this condition is satisfied by all $M \in [M_0]$ so long as $\alpha_{\text{timeout\_retrans}}(\text{queue\_messlist}) = q$ when $M(\text{mess\_channel}) = 1'q$.

   (d) $ATerm(\text{retrans\_counter}, \text{timeout\_retrans}) = \text{rc\_retranscounter}$, and thus
   \[
   Val_{\text{timeout\_retrans}}(ATerm(\text{retrans\_counter}, \text{timeout\_retrans})) = 1'c \text{ for } c \in \text{RetransCounter}.
   \]
   By Lemma 6.1, this condition is satisfied by all $M \in [M_0]$ so long as $\alpha_{\text{timeout\_retrans}}(\text{rc\_retranscounter}) = c$ when $M(\text{retrans\_counter}) = 1'c$.

2. $\forall p \in P \setminus \{\text{sender\_state}, \text{send\_seq\_no}, \text{mess\_channel}, \text{retrans\_counter}\}$,
\[
Val_{\alpha_{\text{timeout\_retrans}}}(p, \text{timeout\_retrans}) = \emptyset, \text{ and } M(p) \geq \emptyset \text{ is true for all } M \in [M_0].
\]

3. $TCond(\text{timeout\_retrans}) = \text{RetransEnabled}_{(\text{retranscounter}, \text{Bool})}(\text{rc\_retranscounter})$, thus
\[
Val_{\text{Bool}, \alpha_{\text{timeout\_retrans}}}(TCond(\text{timeout\_retrans})) = \text{true} \text{ when } \text{RetransEnabled}(c) = \text{true} \text{ for } c \in \text{RetransCounter}, \text{ which implies } c < MR.
\]
By Lemma 6.1, this condition is satisfied by all $M \in [M_0]$ so long as
\[
\alpha_{\text{timeout\_retrans}}(\text{rc\_retranscounter}) = c \text{ when } M(\text{retrans\_counter}) = 1'c \text{ and } c < MR.
\]

Hence $\text{timeout\_retrans}$ is enabled for the set of markings $\{ M \in [M_0] \mid M(\text{sender\_state}) = 1'\text{wait\_ack}, f_c(M(\text{retrans\_counter})) < MR \}$.

Part 2.
Given that \( M \) enables \( \text{timeout}_{\text{retrans}} \), then from the first part of the proof we have\\ \( \alpha_{\text{timeout}_{\text{retrans}}}(s_{\text{seq}}) = i, \alpha_{\text{timeout}_{\text{retrans}}}(q_{\text{ue}_{\text{messlist}}}) = q \) and \( \alpha_{\text{timeout}_{\text{retrans}}}(r_{\text{retranscounter}}) = c \).\n
From \( HLPNG(M_{\text{MS,MR}}) \), \( \text{timeout}_{\text{retrans}} \) has output arcs to \( \text{sender}_{\text{state}}, \text{send}_{\text{seq}_{\text{no}}}, \text{mess}_{\text{channel}} \) and \( \text{retrans}_{\text{counter}} \), where:\n
1. \( A\text{Term}(\text{timeout}_{\text{retrans}}, \text{sender}_{\text{state}}) = \text{wait}_{\text{ack}}_{\text{sender}}, \) and thus\\ \( \text{Val}_{\text{timeout}_{\text{retrans}}}(A\text{Term}(\text{timeout}_{\text{retrans}}, \text{sender}_{\text{state}})) = 1\cdot \text{wait}_{\text{ack}}. \)\n
2. \( A\text{Term}(\text{timeout}_{\text{retrans}}, \text{send}_{\text{seq}_{\text{no}}}) = s_{\text{seq}}, \) and thus\\ \( \text{Val}_{\text{timeout}_{\text{retrans}}}(A\text{Term}(\text{timeout}_{\text{retrans}}, \text{send}_{\text{seq}_{\text{no}}})) = 1\cdot i. \)\n
3. \( A\text{Term}(\text{timeout}_{\text{retrans}}, \text{mess}_{\text{channel}}) = \text{\^{\text{\^{}}}} (m_{\text{esslist}};m_{\text{esslist}};m_{\text{esslist}})(q_{\text{ue}_{\text{messlist}}};m_{\text{esslist}};m_{\text{esslist}}), \) \text{MakeList}(s_{\text{eq}},q_{\text{ue}_{\text{messlist}}};s_{\text{seq}}), \) and thus\\ \( \text{Val}_{\text{timeout}_{\text{retrans}}}(A\text{Term}(\text{timeout}_{\text{retrans}}, \text{mess}_{\text{channel}})) = 1\cdot q^{\text{\^{\text{\^{}}}} [i]} \)\n
4. \( A\text{Term}(\text{timeout}_{\text{retrans}}, \text{retrans}_{\text{counter}}) = \text{IncrementRC}(r_{\text{retranscounter}},r_{\text{retranscounter}})(r_{\text{retranscounter}}), \) and thus\\ \( \text{Val}_{\text{timeout}_{\text{retrans}}}(A\text{Term}(\text{timeout}_{\text{retrans}}, \text{retrans}_{\text{counter}})) = 1\cdot (c + 1). \)\n
5. \( \forall p \in P \setminus \{ \text{sender}_{\text{state}}, \text{send}_{\text{seq}_{\text{no}}}, \text{mess}_{\text{channel}}, \text{retrans}_{\text{counter}} \}, (\text{timeout}_{\text{retrans}}, p) \notin F, \) hence \( \text{Val}_{\text{timeout}_{\text{retrans}}}(\text{timeout}_{\text{retrans}}, p) = \emptyset. \)\n
Hence, from the transition rule:\n
1. \( M'(\text{sender}_{\text{state}}) = M(\text{sender}_{\text{state}}) - 1\cdot \text{wait}_{\text{ack}} + 1\cdot \text{wait}_{\text{ack}} = M(\text{sender}_{\text{state}}) = 1\cdot \text{wait}_{\text{ack}}. \)\n
2. \( M'(\text{send}_{\text{seq}_{\text{no}}}) = M(\text{send}_{\text{seq}_{\text{no}}}) - 1\cdot i + 1\cdot i = M(\text{send}_{\text{seq}_{\text{no}}}) = 1\cdot i. \)\n
3. \( M'(\text{mess}_{\text{channel}}) = M(\text{mess}_{\text{channel}}) - 1\cdot q + 1\cdot q^{\text{\^{\text{\^{}}}} [i]} = 1\cdot q^{\text{\^{\text{\^{}}}} [i]}. \)\n
4. \( M'(\text{retrans}_{\text{counter}}) = M(\text{retrans}_{\text{counter}}) - 1\cdot c + 1\cdot (c + 1) = 1\cdot (c + 1) \)\n\( = 1\cdot (f_{\text{c}}(M(\text{retrans}_{\text{counter}})) + 1). \)\n
5. \( \forall p \in P \setminus \{ \text{sender}_{\text{state}}, \text{send}_{\text{seq}_{\text{no}}}, \text{mess}_{\text{channel}}, \text{retrans}_{\text{counter}} \}, M'(p) = M(p) - \emptyset + \emptyset = M(p). \)\n
Hence, \( \forall M' \in V', M'(\text{mess}_{\text{channel}}) = 1\cdot f_{\text{c}}(M(\text{mess}_{\text{channel}}))^{\text{\^{\text{\^{}}}} [f_{\text{c}}(M(\text{send}_{\text{seq}_{\text{no}}}))]}, M'(\text{retrans}_{\text{counter}}) = 1\cdot (f_{\text{c}}(M(\text{retrans}_{\text{counter}})) + 1), \) and the marking of all other places is unchanged by the occurrence of \( \text{timeout}_{\text{retrans}}. \)

Thus from the proofs of Part 1 and Part 2, the lemma is proved.
Lemma 6.4. For the receive.mess transition:

1. Given $V \in \{ M \mid M \in [M_0] \land |f_c(M(mess\_channel))| > 0 \land M(receiver\_state) = 1'r_{\text{ready}}, M \in V \Rightarrow M[receive.mess]$.  

2. Given $V' = \{ M' \mid M[receive.mess]M' \land M \in V \}, M' \in V' \Rightarrow$
   
   - $M'(receiver\_state) = 1'process$;
   
   - $M'(mess\_channel) = 1'q$ for $M(mess\_channel) = 1'(i :: q), i \in Seq, q \in MessList$;
   
   - $M'(recv\_seq.no) = \begin{cases} 1'(f_c(M(recv\_seq.no)) \oplus MS 1), & \text{if } i = f_c(M(recv\_seq.no)) \\ M(recv\_seq.no), & \text{if } i \neq f_c(M(recv\_seq.no)) \end{cases}$
   
   - $\forall p \in P \setminus \{receiver\_state, mess\_channel, recv\_seq.no\}, M'(p) = M(p)$.


Using the same procedure as the proof of Lemma 6.2, for a particular assignment, $\alpha_{receive.mess}$, to its variables:

1. receive.mess has input arcs from receiver.state, recv_seq.no and mess.channel, where:
   
   (a) $ATerm(receiver\_state, receive.mess) = r_{\text{ready}receiver}$, and thus
   
   $Val_{\alpha_{receive.mess}}(ATerm(receiver\_state, receive.mess)) = 1'r_{\text{ready}}$.
   
   By Lemma 6.1, this condition is satisfied so long as $M(receiver\_state) = 1'r_{\text{ready}}$.

   (b) $ATerm(mess\_channel, receive.mess) = ::(seq.messlist, messlist) (sn_seq, queue.messlist)$,
   
   and thus $Val_{\alpha_{receive.mess}}(ATerm(mess\_channel, receive.mess)) = 1'(i :: q)$ for $i \in Seq, q \in MessList$.
   
   By Lemma 6.1, this condition is satisfied by all $M \in [M_0]$, so long as
   
   $\alpha_{receive.mess}(sn_seq) = i$ and $\alpha_{receive.mess}(queue.messlist) = q$ when $M(mess\_channel) = 1'(i :: q)$, implying that there is at least one element, $i$, in the message channel, i.e.
   
   $|f_c(M(mess\_channel))| > 0$.

   (c) $ATerm(recv\_seq.no, receive.mess) = rn_{seq}$, and thus
   
   $Val_{\alpha_{receive.mess}}(ATerm(recv\_seq.no, receive.mess)) = 1'j$ for $j \in Seq$.
   
   By Lemma 6.1, this condition is satisfied by all $M \in [M_0]$, so long as $\alpha_{receive.mess}(rn_{seq}) = j$ when $M(recv\_seq.no) = 1'j$.

2. $\forall p \in P \setminus \{receiver\_state, recv\_seq.no, mess\_channel\}, Val_{\alpha_{receive.mess}}(p, receive.mess) = \emptyset$,
   
   and $M(p) \geq \emptyset$ is true for all $M \in [M_0]$.

3. $TCond(receive.mess) = true_{\text{Boo}},$ hence $Val_{\text{Boo}, \alpha_{receive.mess}}(TCond(receive.mess)) = true.$
Hence receive\_mess is enabled for the set of markings \( \{ M \in [M_0] \mid M(\text{receiver\_state}) = 1^r\text{\_ready}, |f_c(M(\text{mess\_channel}))| > 0 \} \).

**Part 2.**

Given that \( M \) enables receive\_mess, then from the first part of the proof we have \( \alpha_{\text{receive\_mess}}(\text{sn\_seq}) = i, \quad \alpha_{\text{receive\_mess}}(\text{rn\_seq}) = j \) and \( \alpha_{\text{receive\_mess}}(\text{queue\_messlist}) = q \). From \( HLPNG_{(MS,MR)} \), receive\_mess has output arcs to receiver\_state, mess\_channel and recv\_seq\_no, where:

1. \( ATerm(\text{receive\_mess, receiver\_state}) = \text{process\_receiver} \), and thus
   \[ \text{Val}_{\alpha_{\text{receive\_mess}}}(ATerm(\text{receive\_mess, receiver\_state})) = 1^\text{process} \].

2. \( ATerm(\text{receive\_mess, mess\_channel}) = \text{queue\_messlist} \) and thus
   \[ \text{Val}_{\alpha_{\text{receive\_mess}}}(ATerm(\text{receive\_mess, mess\_channel})) = 1^q \].

3. \( ATerm(\text{receive\_mess, recv\_seq\_no}) = \text{IncrementRSN}_{(\text{seq\_seq, seq})}(\text{sn\_seq, rn\_seq}) \) and thus
   \[ \text{Val}_{\alpha_{\text{receive\_mess}}}(ATerm(\text{receive\_mess, recv\_seq\_no})) = \begin{cases} 1^j \oplus MS \ 1, & \text{if } i = j, \\ 1^j, & \text{if } i \neq j \end{cases} \]

4. \( \forall p \in P \setminus \{ \text{receiver\_state, mess\_channel, recv\_seq\_no} \}, (\text{receive\_mess, } p) \not\in F \), hence \( \text{Val}_{\alpha_{\text{receive\_mess}}} = \emptyset \).

Hence, from the transition rule:

1. \( M'(\text{receiver\_state}) = M(\text{receiver\_state}) - 1^r\text{\_ready} + 1^\text{process} = 1^\text{process} \).

2. \( M'(\text{mess\_channel}) = M(\text{mess\_channel}) - 1^i :: q + 1^q = 1^q \).

3. \( M'(\text{recv\_seq\_no}) = \begin{cases} M(\text{recv\_seq\_no}) - 1^j + 1^j \oplus MS \ 1, & \text{if } i = j, \\ M(\text{recv\_seq\_no}) - 1^j + 1^j = M(\text{recv\_seq\_no}) = 1^j, & \text{if } i \neq j \end{cases} \)

4. \( \forall p \in P \setminus \{ \text{receiver\_state, mess\_channel, recv\_seq\_no} \}, M'(p) = M(p) - \emptyset + \emptyset = M(p) \).

Hence, \( \forall M' \in V', M'(\text{receiver\_state}) = 1^\text{process}, M'(\text{mess\_channel}) = 1^q \) for \( M(\text{mess\_channel}) = 1^i :: q, M'(\text{recv\_seq\_no}) = 1^i(f_c(M(\text{recv\_seq\_no})) \oplus MS \ 1), \) if \( i = f_c(M(\text{recv\_seq\_no})) \), or \( M(\text{recv\_seq\_no}), \) if \( i \neq f_c(M(\text{recv\_seq\_no})) \), and the marking of all other places is unchanged by the firing of receive\_mess.

Thus from the proofs of Part 1 and Part 2, the lemma is proved. \( \square \)

**Lemma 6.5.** For the send\_ack transition:

1. Given \( V = \{ M \mid M \in [M_0] \land M(\text{receiver\_state}) = 1^\text{process} \} \), \( M \in V \implies M[\text{send\_ack}] \).
6.3. Investigating Behavioural Properties

2. Given

\[ V' = \{ M' \mid M[\text{send\_ack}]M' \wedge M \in V \} , \quad M' \in V' \implies \]

- \( M'(\text{receiver\_state}) = 1 \cdot r_{\text{ready}}; \)
- \( M'(\text{ack\_channel}) = 1 \cdot f_c(M(\text{ack\_channel})) \cdot [f_c(M(\text{recv\_seq\_no}))]; \) and
- \( \forall p \in P \setminus \{ \text{receiver\_state}, \text{ack\_channel} \}, M'(p) = M(p). \)

**Proof.** Part 1.
Using the same procedure as the proof of Lemma 6.2, for a particular assignment, \( \alpha_{\text{send\_ack}} \), to its variables:

1. send\_ack has input arcs from receiver\_state, recv\_seq\_no and ack\_channel, where:
   
   (a) \( \text{ATerm(receiver\_state, send\_ack)} = \text{process}_{\text{receiver}}, \) and thus \( \text{Val}_{\alpha_{\text{send\_ack}}} \text{ATerm(receiver\_state, send\_ack)} = 1 \cdot \text{process}. \)
      
      By Lemma 6.1, this condition is satisfied so long as \( M(\text{receiver\_state}) = 1 \cdot \text{process}. \)
   
   (b) \( \text{ATerm(recv\_seq\_no, send\_ack)} = \text{rn\_seq}, \) and thus \( \text{Val}_{\alpha_{\text{send\_ack}}} \text{ATerm(recv\_seq\_no, send\_ack)} = 1 \cdot i \) for \( i \in \text{Seq}. \)
      
      By Lemma 6.1, this condition is satisfied by all \( M \in [M_0] \), so long as \( \alpha_{\text{send\_ack}}(\text{rn\_seq}) = i \) when \( M(\text{recv\_seq\_no}) = 1 \cdot i. \)
   
   (c) \( \text{ATerm(ack\_channel, send\_ack)} = \text{queue\_messlist}, \) and thus \( \text{Val}_{\alpha_{\text{send\_ack}}} \text{ATerm(ack\_channel, send\_ack)} = 1 \cdot q \) for \( q \in \text{MessList}. \)
      
      By Lemma 6.1, this condition is satisfied by all \( M \in [M_0] \), so long as \( \alpha_{\text{send\_ack}}(\text{queue\_messlist}) = q \) when \( M(\text{ack\_channel}) = 1 \cdot q. \)

2. \( \forall p \in P \setminus \{ \text{receiver\_state}, \text{recv\_seq\_no}, \text{ack\_channel} \}, \text{Val}_{\alpha_{\text{send\_ack}}} (p, \text{send\_ack}) = 0, \) and \( M(p) \geq 0 \) is true for all \( M \in [M_0]. \)

3. \( \text{TCond(send\_ack)} = \text{true}_{\text{Bool}}, \) hence \( \text{Val}_{\text{Bool}, \alpha_{\text{send\_ack}}} \text{TCond(send\_ack)} = \text{true}. \)

Hence send\_ack is enabled for the set of markings \( \{ M \in [M_0] \mid M(\text{receiver\_state}) = 1 \cdot \text{process} \}. \)

**Part 2.**

Given that \( M \) enables send\_ack, then from the first part of the proof we have \( \alpha_{\text{send\_ack}}(\text{rn\_seq}) = i \) and \( \alpha_{\text{send\_ack}}(\text{queue\_messlist}) = q. \) From \( \text{HLPNG}_{(MS,MR)} \), send\_ack has output arcs to receiver\_state, recv\_seq\_no and ack\_channel, where:

1. \( \text{ATerm(send\_ack, receiver\_state)} = 1 \cdot r_{\text{ready}_{\text{receiver}}}, \) and thus \( \text{Val}_{\alpha_{\text{send\_ack}}} \text{ATerm(send\_ack, receiver\_state)} = 1 \cdot r_{\text{ready}}. \)
6.3. Investigating Behavioural Properties

2. \(\text{ATerm}(\text{send\_ack}, \text{recv\_seq\_no}) = r_{\text{seq}}\), and thus
   \(\text{Val}_{\text{send\_ack}}(\text{ATerm}(\text{send\_ack}, \text{recv\_seq\_no})) = 1^i\).

3. \(\text{ATerm}(\text{send\_ack}, \text{ack\_channel}) = \text{^\_messlist.messlist.messlist}(\text{queue.messlist}),\)
   \(\text{MakeList}(\text{seq.messlist})(r_{\text{seq}})\), and thus
   \(\text{Val}_{\text{send\_ack}}(\text{ATerm}(\text{send\_ack}, \text{ack\_channel})) = 1^q[i]\).

4. \(\forall p \in P \setminus \{\text{receiver\_state}, \text{recv\_seq\_no}, \text{ack\_channel}\}, (\text{send\_ack}, p) \notin F\), hence
   \(\text{Val}_{\text{send\_ack}}(\text{send\_ack}, p) = \emptyset\).

Hence, from the transition rule:

1. \(M'(\text{receiver\_state}) = M(\text{receiver\_state}) - 1^r_{\text{process}} + 1^r_{\text{ready}} = 1^r_{\text{ready}}\).

2. \(M'(\text{recv\_seq\_no}) = M(\text{recv\_seq\_no}) - 1^i + 1^i = M(\text{recv\_seq\_no}) = 1^i\).

3. \(M'(\text{ack\_channel}) = M(\text{ack\_channel}) - 1^q + 1^q[i] = 1^q[i]\).

4. \(\forall p \in P \setminus \{\text{receiver\_state}, \text{recv\_seq\_no}, \text{ack\_channel}\}, M'(p) = M(p) - \emptyset + \emptyset = M(p)\).

Hence, \(\forall M' \in V'\), \(M'(\text{receiver\_state}) = 1^r_{\text{ready}}, M'(\text{ack\_channel}) = 1^f_c(M(\text{ack\_channel}))^{-}[f_c(M(\text{recv\_seq\_no}))]\), and the marking of all other places is unchanged by the occurrence of \text{send\_ack}.

Thus from the proofs of Part 1 and Part 2, the lemma is proved.

\[\square\]

**Lemma 6.6.** For the \text{receive\_ack} transition:

1. Given \(V = \{M \mid M \in [M_0) \land M(\text{sender\_state}) = 1^\text{wait\_ack} \land [f_c(M(\text{ack\_channel}))] > 0 \land f_c(M(\text{send\_seq\_no})) \oplus MS 1 = r_{\text{seq}}, \text{where } (r_{\text{seq}} :: \text{queue}) = f_c(M(\text{ack\_channel}))\},\)
   \(M \in V \implies M[\text{receive\_ack}]\).

2. Given \(V' = \{M' \mid M[\text{receive\_ack}]M' \land M \in V\}, M' \in V' \implies\)

   - \(M'(\text{sender\_state}) = 1^s_{\text{ready}};\)
   - \(M'(\text{send\_seq\_no}) = 1^j(f_c(M(\text{send\_seq\_no})) \oplus MS 1);\)
   - \(M'(\text{ack\_channel}) = 1^q\text{ for } M(\text{ack\_channel}) = 1^j(q :: j) \in \text{Seq}, q \in \text{MessList};\)
   - \(M'(\text{retrans\_counter}) = 1^0;\) and
   - \(\forall p \in P \setminus \{\text{sender\_state}, \text{send\_seq\_no}, \text{retrans\_counter}, \text{ack\_channel}\}, M'(p) = M(p).\)

**Proof.** Part 1.

Using the same procedure as in the proof of Lemma 6.2, for a particular assignment, \(\alpha_{\text{receive\_ack}}\), to its variables:
6.3. Investigating Behavioural Properties

1. receive_ack has input arcs from sender_state, send_seq_no, ack_channel and retrans_counter, where:

(a) \(ATerm(sender_state, receive_ack) = wait\_ack\_sender\), and thus
\[Val_{\alpha_{receive\_ack}}(ATerm(sender_state, receive\_ack)) = 1'wait\_ack\].

By Lemma 6.1, this condition is satisfied so long as \(M(sender_state) = 1'wait\_ack\).

(b) \(ATerm(send\_seq\_no, receive\_ack) = sn_{seq}\), and thus
\[Val_{\alpha_{receive\_ack}}(ATerm(send\_seq\_no, receive\_ack)) = 1'i\] for \(i \in Seq\).

By Lemma 6.1, this condition is satisfied by all \(M \in \{M_0\}\), so long as \(\alpha_{receive\_ack}(sn_{seq}) = i\) when \(M(send\_seq\_no) = 1'i\).

(c) \(ATerm(ack\_channel, receive\_ack) = send\_seq\_list\) (\(rn_{seq}, queue\_messlist\)), and so
\[Val_{\alpha_{receive\_ack}}(ATerm(ack\_channel, receive\_ack)) = 1'(j :: q)\] for \(j \in Seq, q \in MessList\).

By Lemma 6.1, this condition is satisfied by all \(M \in \{M_0\}\), so long as
\(\alpha_{receive\_ack}(rn_{seq}) = j\) and \(\alpha_{receive\_ack}(queue\_messlist) = q\) when \(M(ack\_channel) = 1'(j :: q)\), implying that there is at least one element, \(j\), in the acknowledgement channel, i.e. \(|f_c(M(ack\_channel))| > 0\).

(d) \(ATerm(retrans\_counter, receive\_ack) = retrans\_counter\), and thus
\[Val_{\alpha_{receive\_ack}}(ATerm(retrans\_counter, receive\_ack)) = 1'c\] for \(c \in RetransCounter\).

By Lemma 6.1, this condition is satisfied by all \(M \in \{M_0\}\), so long as
\(\alpha_{receive\_ack}(rc_{retrans\_counter}) = c\) when \(M(retrans\_counter) = 1'c\).

2. \(\forall p \in P \setminus \{sender\_state, send\_seq\_no, ack\_channel, retrans\_counter\}\),
\[Val_{\alpha_{receive\_ack}}(p, receive\_ack) = \emptyset,\] and \(M(p) \geq \emptyset\) is true for all \(M \in \{M_0\}\).

3. \(TCond(receive\_ack)\) equals the term \(=(seq\_seq,Bool) (rn_{seq}, NextSeq(seq,seq)(sn_{seq}))\), hence
\[Val_{Bool,\alpha_{receive\_ack}}(TCond(receive\_ack)) = true\] when \(j = i \oplus MS 1\) for \(i, j \in Seq\).

By Lemma 6.1, this condition is satisfied by all \(M \in \{M_0\}\) so long as \(\alpha_{receive\_ack}(sn_{seq}) = i\) and \(\alpha_{receive\_ack}(rn_{seq}) = j\) when \(M(ack\_channel) = 1'(j :: q)\), implying \(|f_c(M(ack\_channel))| > 0\), and \(M(send\_seq\_no) = 1'i\), such that \(j = i \oplus MS 1\).

Hence receive_ack is enabled for the set of markings \(\{M \in \{M_0\} \mid M(sender\_state) = 1'wait\_ack, |f_c(M(ack\_channel))| > 0, f_c(M(send\_seq\_no)) \oplus MS 1 = rn\}\), where \((rn :: queue) = f_c(M(ack\_channel))\).

Part 2.

Given \(M\) enables receive_ack, then from the first part of this proof we have \(\alpha_{receive\_ack}(sn_{seq}) = i, \alpha_{receive\_ack}(rn_{seq}) = j, \alpha_{receive\_ack}(queue\_messlist) = q,\) and \(\alpha_{receive\_ack}(rc_{retrans\_counter}) = c\). From
6.3. Investigating Behavioural Properties

HLPNG\(_{(MS,MR)}\), receive\_ack has output arcs to sender\_state, send\_seq\_no, ack\_channel and retrans\_counter, where:

1. \(\text{ATerm}(\text{receive\_ack}, \text{sender\_state}) = s\_ready\_sender\), and thus
   \(\text{Val}_{\text{receive\_ack}}(\text{ATerm}(\text{receive\_ack}, \text{sender\_state})) = 1's\_ready\).

2. \(\text{ATerm}(\text{receive\_ack}, \text{send\_seq\_no}) = \text{NextSeq}(\text{seq}, \text{seq})(\text{sn\_seq})\), and thus
   \(\text{Val}_{\text{receive\_ack}}(\text{ATerm}(\text{receive\_ack}, \text{send\_seq\_no})) = 1'i \oplus MS 1\).

3. \(\text{ATerm}(\text{receive\_ack}, \text{ack\_channel}) = \text{queue\_messlist}\), and thus
   \(\text{Val}_{\text{receive\_ack}}(\text{ATerm}(\text{receive\_ack}, \text{ack\_channel})) = 1'q\).

4. \(\text{ATerm}(\text{receive\_ack}, \text{retrans\_counter}) = 0\_\text{retrans\_counter}\), and thus
   \(\text{Val}_{\text{receive\_ack}}() = 1'0\).

5. \(\forall p \in P \setminus \{\text{sender\_state}, \text{send\_seq\_no}, \text{ack\_channel}, \text{retrans\_counter}\}, (\text{receive\_ack}, p) \notin F\),
   hence \(\text{Val}_{\text{receive\_ack}}(\text{receive\_ack}, p) = \emptyset\).

Hence, from the transition rule:

1. \(M'(\text{sender\_state}) = M(\text{sender\_state}) - 1'\text{wait\_ack} + 1's\_ready = 1's\_ready\).

2. \(M'(\text{send\_seq\_no}) = M(\text{send\_seq\_no}) - 1'i + 1'(i \oplus MS 1) = 1'(i \oplus MS 1)\).

3. \(M'(\text{ack\_channel}) = M(\text{ack\_channel}) - 1'(j :: q) + 1'q = 1'q\).

4. \(M'(\text{retrans\_counter}) = M(\text{retrans\_counter}) - 1'c + 1'0 = 1'0\).

5. \(\forall p \in P \setminus \{\text{sender\_state}, \text{send\_seq\_no}, \text{ack\_channel}, \text{retrans\_counter}\}, M'(p) = M(p) - \emptyset + \emptyset = M(p)\).

Hence \(\forall M' \in V', M'(\text{sender\_state}) = 1's\_ready, M'(\text{send\_seq\_no}) = 1'(f_c(M(\text{send\_seq\_no})) \oplus MS 1), M'(\text{ack\_channel}) = 1'q\) for \(M(\text{ack\_channel}) = 1'(j :: q), j \in Seq, M'(\text{retrans\_counter}) = 1'0\),
and the marking of all other places is unchanged by the firing of receive\_ack.

Thus from the proofs of Part 1 and Part 2, the lemma is proved. \[\Box\]

**Lemma 6.7.** For the receive\_dup\_ack transition:

1. Given \(V = \{M \mid M \in [M_0] \land |f_c(M(\text{ack\_channel}))| > 0 \land f_c(M(\text{send\_seq\_no})) \oplus MS 1 \neq rn, \text{ where } (rn :: \text{queue}) = f_c(M(\text{ack\_channel}))\}, M \in V \implies M[\text{receive\_dup\_ack}]\).

2. Given \(V' = \{M' \mid M[\text{receive\_dup\_ack}]M' \land M \in V\}, M' \in V' \implies \)
6.3. Investigating Behavioural Properties

- \( M'(\text{ack\_channel}) = 1'q \) for \( M(\text{ack\_channel}) = 1'(i :: q), i \in \text{Seq}, q \in \text{MessList} \); and
- \( \forall p \in P \setminus \{\text{ack\_channel}\}, M'(p) = M(p) \).

**Proof.** Part 1.

Using the same procedure as in the proof of Lemma 6.2, for a particular assignment, \( \alpha_{\text{receive\_dup\_ack}} \), to its variables:

1. \( \text{receive\_dup\_ack} \) has input arcs from \( \text{send\_seq\_no} \), and \( \text{ack\_channel} \), where:
   
   (a) \( A\text{Term}(\text{send\_seq\_no}, \text{receive\_dup\_ack}) = sn_{\text{seq}}, \) and thus \( \text{Val}_{\alpha_{\text{receive\_dup\_ack}}}(A\text{Term}(\text{send\_seq\_no}, \text{receive\_dup\_ack})) = 1'i \) for \( i \in \text{Seq} \).
   
   By Lemma 6.1, this condition is satisfied by all \( M \in [M_0] \), so long as
   
   \( \alpha_{\text{receive\_dup\_ack}}(sn_{\text{seq}}) = i \) when \( M(\text{send\_seq\_no}) = 1'i. \)

   (b) \( A\text{Term}(\text{ack\_channel}, \text{receive\_dup\_ack}) = \llparentheses \text{seq\_messlist}, \text{messlist} \rrparentheses (rn_{\text{seq}}, \text{queue\_messlist}), \) and thus \( \text{Val}_{\alpha_{\text{receive\_dup\_ack}}}(A\text{Term}(\text{ack\_channel}, \text{receive\_dup\_ack})) = 1'(j :: q) \) for \( j \in \text{Seq}, q \in \text{MessList} \).
   
   By Lemma 6.1, this condition is satisfied by all \( M \in [M_0] \), so long as
   
   \( \alpha_{\text{receive\_dup\_ack}}(rn_{\text{seq}}) = j \) and \( \alpha_{\text{receive\_dup\_ack}}(\text{queue\_messlist}) = q \) when \( M(\text{ack\_channel}) = 1'(j :: q) \), implying that there is at least one element, \( j \), in the acknowledgement channel, i.e.
   
   \( |f_c(M(\text{ack\_channel}))| > 0. \)

2. \( \forall p \in P \setminus \{\text{send\_seq\_no}, \text{ack\_channel}\}, \text{Val}_{\alpha_{\text{receive\_dup\_ack}}}(p, \text{receive\_dup\_ack}) = \emptyset, \) and \( M(p) \geq 0 \) is true for all \( M \in [M_0] \).

3. \( T\text{Cond}(\text{receive\_dup\_ack}) = \llangle \text{seq\_seq}, \text{Bool} \rrangle (rn_{\text{seq}}, \text{NextSeq}(\text{seq}, \text{seq})(sn_{\text{seq}})), \) hence
   
   \( \text{Val}_{\text{Boo}, \alpha_{\text{receive\_dup\_ack}}}(T\text{Cond}(\text{receive\_dup\_ack})) = \text{true} \) when \( j \neq i \oplus_{\text{MS}} 1 \) for \( i, j \in \text{Seq} \). By Lemma 6.1, this condition is satisfied by all \( M \in [M_0] \) so long as \( \alpha_{\text{receive\_dup\_ack}}(sn_{\text{seq}}) = i \) and
   
   \( \alpha_{\text{receive\_dup\_ack}}(rn_{\text{seq}}) = j \) when \( M(\text{ack\_channel}) = 1'(j :: q) \), implying \( |f_c(M(\text{ack\_channel}))| > 0 \) and \( M(\text{send\_seq\_no}) = 1'i, \) such that \( j \neq i \oplus_{\text{MS}} 1. \)

Hence \( \text{receive\_dup\_ack} \) is enabled for the set of markings \( \{M \in [M_0] \mid |f_c(M(\text{ack\_channel}))| > 0, f_c(M(\text{send\_seq\_no})) \oplus_{\text{MS}} 1 = rn, \) where \( (rn :: queue) = f_c(M(\text{ack\_channel})) \} \).

**Part 2.**

Given \( M \) enables \( \text{receive\_dup\_ack} \), then from the first part of this proof we have \( \alpha_{\text{receive\_dup\_ack}}(sn_{\text{seq}}) = i, \alpha_{\text{receive\_dup\_ack}}(rn_{\text{seq}}) = j, \) and \( \alpha_{\text{receive\_dup\_ack}}(\text{queue\_messlist}) = q \). From \( HLPNG(MS, MR) \), \( \text{receive\_dup\_ack} \) has output arcs to \( \text{send\_seq\_no} \) and \( \text{ack\_channel} \), where:
6.3. Investigating Behavioural Properties

1. \( ATerm(\text{receive}\_\text{dup}_\text{ack}, \text{send}_\text{seq}\_\text{no}) = sn_{seq} \), and thus
\( Val_{\text{receive}_\text{dup}_\text{ack}}(ATerm(\text{receive}\_\text{dup}_\text{ack}, \text{send}_\text{seq}\_\text{no})) = 1'\).

2. \( ATerm(\text{receive}\_\text{dup}_\text{ack}, \text{ack}_\text{channel}) = queue_{\text{messlist}} \), and thus
\( Val_{\text{receive}_\text{dup}_\text{ack}}(ATerm(\text{receive}\_\text{dup}_\text{ack}, \text{ack}_\text{channel})) = 1'q.\)

3. \( 8p \in P \setminus \{\text{send}_\text{seq}\_\text{no}, \text{ack}_\text{channel}\}, (\text{receive}_\text{dup}_\text{ack}, p) \notin F \), hence
\( Val_{\text{receive}_\text{dup}_\text{ack}}(\text{receive}_\text{dup}_\text{ack}, p) = 0.\)

Hence, from the transition rule:

1. \( M'(\text{send}_\text{seq}\_\text{no}) = M(\text{send}_\text{seq}\_\text{no}) - 1'i + 1'i = M(\text{send}_\text{seq}\_\text{no}) = 1'i.\)

2. \( M'(\text{ack}_\text{channel}) = M(\text{ack}_\text{channel}) - 1'(j :: q) + 1'q = 1'q.\)

3. \( \forall p \in P \setminus \{\text{send}_\text{seq}\_\text{no}, \text{ack}_\text{channel}\}, M'(p) = M(p) - 0 + 0 = M(p).\)

Hence \( \forall M' \in V', M'(\text{ack}_\text{channel}) = 1'q \) for \( M(\text{ack}_\text{channel}) = 1'(j :: q), j \in \text{Seq} \), and the marking of all other places is unchanged by the occurrence of \( \text{receive}_\text{dup}_\text{ack}.\)

Thus from the proofs of Part 1 and Part 2, the lemma is proved.

Lemma 6.8. **For the mess\_loss transition:**

1. Given \( V = \{ M \mid M \in [M_0) \land f_c(M(\text{mess}_\text{channel})) > 0 \}, M \in V \implies M(\text{mess}_\text{loss}).\)

2. Given \( V' = \{ M' \mid M(\text{mess}_\text{loss})M' \land M \in V \}, M' \in V' \implies \)

   - \( M'(\text{mess}_\text{channel}) = 1'q_1 \sim q_2 \) for \( M(\text{mess}_\text{channel}) = 1'q_1 \sim [i] \sim q_2, i \in \text{Seq},\)
   - \( q_1, q_2 \in \text{MessList}; \) and
   - \( \forall p \in P \setminus \{\text{mess}_\text{channel}\}, M'(p) = M(p).\)


Using the same procedure as in the proof of Lemma 6.2, for a particular assignment, \( \alpha_{\text{mess}_\text{loss}}, \) to its variables:

1. \( \text{mess}_\text{loss} \) has an input arc from \( \text{mess}_\text{channel} \) only, where:

   a) \( ATerm(\text{mess}_\text{channel}, \text{mess}_\text{loss}) = queue_{\text{messlist}} \), and thus
   \( Val_{\text{mess}_\text{loss}}(ATerm(\text{mess}_\text{channel}, \text{mess}_\text{loss})) = 1'q \) for \( q \in \text{MessList}.\)

   By Lemma 6.1, this condition is satisfied by all \( M \in [M_0), \) so long as
   \( \alpha_{\text{mess}_\text{loss}}(queue_{\text{messlist}}) = q \) when \( M(\text{mess}_\text{channel}) = 1'q.\)
2. \( \forall p \in P \setminus \{\text{mess\_channel}\}, Val_{\alpha_{\text{mess\_loss}}}^{\text{mess\_loss}}(p, \text{mess\_loss}) = \emptyset \), and \( M(p) \geq 0 \) is true for all \( M \in [M_0] \).

3. \( T\text{Cond}(\text{mess\_loss}) = \text{Contains}_{(\text{mess\_list.seq, Bool})}(\text{queue\_mess\_list}, sn_{seq}) \), hence

\[
Val_{\alpha_{\text{mess\_loss}}}^{\text{mess\_loss}}(T\text{Cond}(\text{mess\_loss})) = \text{true} \text{ when } \text{Contains}(q, i) = \text{true}, \text{ for } queue \in \text{MessList}, i \in Seq. \]

By Lemma 6.1, this condition is satisfied by all \( M \in [M_0] \) so long as \( \alpha_{\text{mess\_loss}}(\text{queue\_mess\_list}) = q \) and \( \alpha_{\text{mess\_loss}}(sn_{seq}) = i \) when \( M(\text{mess\_channel}) = 1'q \) such that \( q \) contains at least one instance of \( i \), implying that \( |q| > 0 \), that there is at least one element in the message channel, i.e. \( |f_c(M(\text{mess\_channel}))| > 0 \).

Hence \( \text{mess\_loss} \) is enabled for the set of markings \( \{M \in [M_0] \mid |f_c(M(\text{mess\_channel}))| > 0\} \).

**Part 2.**

Given \( M \) enables \( \text{mess\_loss} \), then from the first part of the proof we have \( \alpha_{\text{mess\_loss}}(\text{queue\_mess\_list}) = q \) and \( \alpha_{\text{mess\_loss}}(sn_{seq}) = i \). From \( HLPNG_{(MS, MR)} \), \( \text{mess\_loss} \) has an output arc to \( \text{mess\_channel} \) only, where:

1. \( A\text{Term}(\text{mess\_loss, mess\_channel}) = L\text{oss}_{(\text{mess\_list.seq, mess\_list})}(\text{queue\_mess\_list}, sn_{seq}) \), and thus
   \[
   Val_{\alpha_{\text{mess\_loss}}}^{\text{mess\_loss}}(A\text{Term}(\text{mess\_loss, mess\_channel})) = 1'q_1 \cdots q_2 \text{ for } q = q_1 \cdots [i] \cdots q_2.
   \]

2. \( \forall p \in P \setminus \{\text{mess\_channel}\}, (\text{mess\_loss, p}) \notin F \), hence \( Val_{\alpha_{\text{mess\_loss}}}^{\text{mess\_loss}}(\text{mess\_loss, p}) = \emptyset \).

Hence, from the transition rule:

1. \( M'(\text{mess\_channel}) = M(\text{mess\_channel}) - 1'q + 1'q_1 \cdots q_2 = 1'q_1 \cdots q_2 \).

2. \( \forall p \in P \setminus \{\text{mess\_loss}\}, M'(p) = M(p) - \emptyset + \emptyset = M(p) \).

Hence, \( \forall M' \in V', M'(\text{mess\_channel}) = 1'q_1 \cdots q_2 \) for \( M(\text{mess\_channel}) = 1'q_1 \cdots [i] \cdots q_2, i \in Seq \), and the marking of all other places is unchanged by the occurrence of \( \text{mess\_loss} \).

Thus the lemma is proved.

**Lemma 6.9.** For the \( \text{ack\_loss} \) transition:

1. Given \( V = \{M \mid M \in [M_0] \land |f_c(M(\text{ack\_channel}))| > 0\}, M \in V \implies M[\text{ack\_loss}] \).

2. Given \( V' = \{M' \mid M[\text{ack\_loss}]M' \land M \in V\}, M' \in V' \implies \)

   - \( M'(\text{ack\_channel}) = 1'q_1 \cdots q_2 \) for \( M(\text{ack\_channel}) = 1'q_1 \cdots [i] \cdots q_2, i \in Seq, q_1, q_2 \in \text{MessList} \); and
   - \( \forall p \in P \setminus \{\text{ack\_channel}\}, M'(p) = M(p) \).
6.4 Concluding Remarks

Proof. The proof of this lemma is identical to the proof of Lemma 6.8 when $\text{ack}_{\text{loss}}$, $\text{ack}_{\text{channel}}$ and $rn_{\text{seq}}$ are substituted for $\text{mess}_{\text{loss}}$, $\text{mess}_{\text{channel}}$ and $sn_{\text{seq}}$, respectively.

Lemmas 6.2 to 6.9 present key results, as they allow a generic notation to be defined for markings and arcs for all values of $i$, $0 \leq i \leq MS$. They are also critical in the proof of correctness of the algebraic expressions of the Parameterised Reachability Graph presented in Chapter 8, and the parametric analysis carried out in Chapters 9 and 10. They eliminate the need for lengthy and repetitive proofs by induction (or similar) over the MaxSeqNo parameter.

6.4 Concluding Remarks

Our preliminary investigations into the parameterised Stop-and-Wait Protocol model are encouraging. When there are no retransmissions, we have been able to identify repeating patterns in concrete reachability graphs for small values of the MaxSeqNo parameter, formalise a parametric reachability graph in MaxSeqNo, and conduct analysis directly on this parametric reachability graph (see [48, 49]). When considering retransmissions, we have been able to derive an expression for the size of the reachability graph that is factorable in both parameters, from data on the size of concrete reachability graphs over a range of parameter values. The fact that the size is factorable into the two parameters gives a strong indication that similar repeating patterns also exist in the reachability graphs over both parameters. We have examined the behaviour of the SWP CPN model and found that exactly one token will mark each place in every reachable marking. We have also been able to prove from the HLPNG representation that the enabling and occurrence of every transition is ‘independent’ of the sequence numbers involved. The next chapter takes advantage of these two properties and defines a shorthand notation that we can use to identify markings and arcs in a parametric reachability graph.
Chapter 7

A Notation for Markings and Arcs

In this chapter a notation is defined for the markings and arcs of our parametric reachability graph. This chapter presents a substantially more mature version of the work originally presented in [50, 51]. We begin in Section 7.1 by introducing the notational conventions used in the marking and arc notation developed in Section 7.2. In Section 7.3 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 7.4. Finally, we establish formulas for determining the queue lengths (the number of messages and acknowledgements in the channels) and special sets of markings called downward-closed sets, directly from the parametric notation, in Sections 7.5 and 7.6, respectively.

7.1 Notational Conventions

Recall from the start of Section 5.5 that $MS$ and $MR$ are used as shorthand for the parameters MaxSeqNo and MaxRetrans respectively. In addition, the following notational conventions are used throughout the remainder of this thesis:

- $i^j$ 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 (from Chapter 5); and
- $\ominus_{MS}$ is used to represent modulo $MS + 1$ subtraction.

7.2 Marking and Arc Notation

Given Lemma 6.1, 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.
7.2. Marking and Arc Notation

7.2.1 Marking Notation

Recall from Chapter 2 the definition of a marking of a HLPNG (Definition 2.10):\(^1\)

\[ M : P \rightarrow \bigcup_{p \in P} \mu\text{Type}(p) \text{ such that } \forall p \in P, M(p) \in \mu\text{Type}(p) \]

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

\[ M = \{(p, mp) \mid p \in P\}, \text{ where } mp \in \mu\text{Type}(p) \]

For the case where \(|mp| = 1\), i.e. \(mp = 1'g\) where \(g \in \text{Type}(p)\), 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 6.1 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:

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

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:

\[ M = (M(\text{sender\_state}), M(\text{receiver\_state}), M(\text{send\_seq\_no}), M(\text{recv\_seq\_no}), \]

\[ M(\text{mess\_channel}), M(\text{ack\_channel}), M(\text{retrans\_counter})) \]

Because of Lemma 6.1, \(M(\text{sender\_state}), M(\text{receiver\_state}), M(\text{send\_seq\_no}), M(\text{recv\_seq\_no}), \) 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:

\[ s\_state \in \text{Sender}, \]
\[ r\_state \in \text{Receiver}, \]
\[ ssn \in \text{Seq} \]
\[ rsn \in \text{Seq} \]
\[ \{ \text{as in Section 6.1, and} \]
\[ ret \in \text{RetransCounter}, \text{ respectively.} \]

The marking of the message channel, \(M(\text{mess\_channel})\), and acknowledgement channel, \(M(\text{ack\_channel})\), cannot be represented as easily. In [19, 20] the marking of the message and acknowledgement channels were each encoded into four integer variables. In this thesis we develop 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

\(^1\)With thanks to J. Billington for suggesting a formal derivation of the marking notation from the marking of a HLPNG.
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 7.1.** 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 message 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, \((\text{MaxRetrans+1}) \ (n \oplus_{MS} 1)\)-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.

Similarly, for the acknowledgement channel:

**Lemma 7.2.** The content of the acknowledgement channel can be represented by:

\[
M(\text{ack\_channel}) = 1^{[ssn^{ao}(ssn \ominus_{MS} 1)^{an}}
\]

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 7.1, except that we are dealing with acknowledgements in the acknowledgement channel, not messages in the message channel, with sequence numbers of \(ssn\) and \(ssn \ominus_{MS} 1\) instead of \(ssn \ominus_{MS} 1\) and \(ssn\), respectively.
7.2. Marking and Arc Notation

As we will see in Chapter 8, although this representation can express only a small subset of the elements in \textit{MessList}, it turns out that this representation is sufficiently expressive to capture the channel content of all reachable markings. Thus from Equation (7.1), the marking of the net can be represented by the vector

\[ M = (s\_state, r\_state, ssn, rsn, [(ssn \oplus MS 1)^{mo} ssn^{mn}], [ssn^{ao}, (ssn \oplus MS 1)^{an}], ret) \]

and hence we can see that a marking of the CPN can be characterised by 9 variables, plus the two parameters, because \textit{ret} is restricted by \textit{MR}.

We are now ready to define our marking notation.

**Definition 7.1** (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. 5.5 and 5.6, where the superscript is a pair of the parameter values of the SWP CPN and the subscript encodes the marking description, such that:

- \textit{s\_state} \in Sender is the state of the Sender, i.e. either ready or waiting for an acknowledgement;
- \textit{r\_state} \in Receiver is the state of the Receiver, i.e. either ready or processing a message;
- \textit{ssn} \in Seq is the sender sequence number;
- \textit{rsn} \in Seq is the receiver sequence number;
- \textit{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, \( ssn \oplus MS 1 \)) in the message channel;
- \textit{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;
- \textit{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;
- \textit{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
7.2. Marking and Arc Notation

- \( \text{ret} \in \text{RetransCounter} \) is the number of times the currently outstanding message has been retransmitted;

where the marking of each place is given by:

\[
\begin{align*}
M(\text{sender \_state}) &= 1's_{\text{state}} \\
M(\text{send \_seq\_no}) &= 1'ssn \\
M(\text{retrans\_counter}) &= 1'ret \\
M(\text{mess\_channel}) &= 1'[ssn \oplus_{MS} 1]^{1\text{mo}} ssn^{mn} \\
M(\text{ack\_channel}) &= 1'[ssn^{ao} (ssn \oplus_{MS} 1)^{1\text{an}}]
\end{align*}
\]

In Definition 7.1 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 \( M(3;4) \) (\text{wait \_ack};r \_ready;2,2). This corresponds to the marking of \( CPN(3;4) \):

\[
\begin{align*}
M(\text{sender \_state}) &= 1'\text{wait \_ack} \\
M(\text{sender \_seq\_no}) &= 1'2 \\
M(\text{mess\_channel}) &= 1'[1,1,2,2] \\
M(\text{ack\_channel}) &= 1'[2]
\end{align*}
\]

where \( MS = 3 \) and \( MR = 4 \).

### 7.2.2 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 2.14 and the transition rule in Definition 2.12, 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 \text{send \_mess} transition and a marking,
that enables \texttt{send\_mess}. From the proof of Lemma 6.2, \texttt{send\_mess}
will only be enabled in this marking if the variable \texttt{sn} is bound to 0 and \texttt{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, \( M = M^{(\text{MS}, \text{MR})}_{(s\_\text{state}, r\_\text{state}), (\text{ssn}, \text{rsn}), (\text{mo}, \text{ao}, \text{mn}, \text{an}, \text{ret})} \), such that \( M[t] \) \( (t \text{ is enabled in } M) \):

- From Lemma 6.2, \( M[\text{send\_mess}] \) with the binding \( (\text{queue} = [(\text{ssn } \oplus \text{MS } 1)^{\text{mo}} \text{ ssn}^{\text{mn}}], \text{sn} = \text{ssn}) \).

- From Lemma 6.3, \( M[\text{timeout\_retrans}] \) with the binding \( (\text{queue} = [(\text{ssn } \oplus \text{MS } 1)^{\text{mo}} \text{ ssn}^{\text{mn}}], \text{sn} = \text{ssn}, rc = \text{ret}) \), with \( rc < \text{MR} \).

- From Lemma 6.4, \( M[\text{receive\_mess}] \) with the binding:
  
  \[-(\text{queue} = [(\text{ssn } \oplus \text{MS } 1)^{\text{mo}-1} \text{ ssn}^{\text{mn}}], \text{sn} = \text{ssn } \oplus \text{MS } 1, \text{rn} = \text{rsn}), \text{if } \text{mo} > 0 \text{; or}\]
  
  \[-(\text{queue} = [\text{ssn}^{\text{mn}-1}], \text{sn} = \text{ssn}, \text{rn} = \text{rsn}), \text{if } \text{mo} = 0 \text{ and } \text{mn} > 0.\]

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.

- From Lemma 6.5, \( M[\text{send\_ack}] \) with the binding \( (\text{queue} = [\text{ssn}^{\text{ao}} (\text{ssn } \oplus \text{MS } 1)^{\text{an}}], \text{rn} = \text{rsn}) \).

- From Lemma 6.6, \( M[\text{receive\_ack}] \) with the binding \( (\text{queue} = [(\text{ssn } \oplus \text{MS } 1)^{\text{an}-1}], \text{sn} = \text{ssn}, \text{rn} = \text{ssn } \oplus \text{MS } 1, rc = \text{ret}) \). Recall that \( rc \) must equal \( \text{sn } \oplus \text{MS } 1 \) from the guard.

The guard prevents this transition from being enabled if there are any old acknowledgements in
the channel (i.e. \( \text{ao} \) must equal zero) because our marking notation specifies that old acknowledgements
come before new acknowledgements in the (FIFO) channel.

- From Lemma 6.7, \( M[\text{receive\_dup\_ack}] \) with the binding \( (\text{queue} = [\text{ssn}^{\text{ao}-1} (\text{ssn } \oplus \text{MS } 1)^{\text{an}}], \text{sn} = \text{ssn}, \text{rn} = \text{ssn}) \). Recall that \( \text{rn} \) cannot equal \( \text{sn } \oplus \text{MS } 1 \) from the guard. Our marking
notation specifies that old acknowledgments have sequence number \( \text{ssn} \) and that the number of
old acknowledgements is given by \( \text{ao} \). Hence, \( \text{ao} \) must be greater than zero for this transition to be
enabled.

- From Lemma 6.8, \( M[\text{mess\_loss}] \) with the binding:
  
  \[-(\text{queue} = [(\text{ssn } \oplus \text{MS } 1)^{\text{mo}} \text{ ssn}^{\text{mn}}], \text{sn} = \text{ssn } \oplus \text{MS } 1), \text{provided } \text{mo} > 0 \text{; or}\]
  
  \[-(\text{queue} = [(\text{ssn } \oplus \text{MS } 1)^{\text{mo}} \text{ ssn}^{\text{mn}}], \text{sn} = \text{ssn}), \text{provided } mn > 0.\]
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.

- From Lemma 6.9, \(M[\text{ack\,loss}]\) with the binding:
  
  \[- (queue = [ssn^{ao} (ssn + MS 1)^{an}], rn = ssn), \text{provided } ao > 0; \text{ or} \]
  
  \[- (queue = [ssn^{ao} (ssn + MS 1)^{an}], rn = ssn + MS 1), \text{provided } an > 0. \]

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:

\[
ATNames = \{\text{send\,mess, timeout\,retrans, receive\,mess, send\,ack, receive\,ack, receive\,dup\,ack, mess\,loss\,old, mess\,loss\,new, ack\,loss\,old, ack\,loss\,new}\}
\]

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 7.2 (Augmented Transition Name to Transition Name).**

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

\[
\text{TransMap}(\text{atn}) = \begin{cases} 
\text{atn}, & \text{for } \text{atn} \in \{\text{send}_\text{mess}, \text{timeout}_\text{retrans}, \text{receive}_\text{mess}, \text{send}_\text{ack}, \\
\text{receive}_\text{ack}, \text{receive}_\text{dup}_\text{ack}\}; \\
\text{mess}_\text{loss}, & \text{for } \text{atn} \in \{\text{mess}_\text{loss}_\text{old}, \text{mess}_\text{loss}_\text{new}\}, \\
\text{ack}_\text{loss}, & \text{for } \text{atn} \in \{\text{ack}_\text{loss}_\text{old}, \text{ack}_\text{loss}_\text{new}\}.
\end{cases}
\]

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

**Definition 7.3 (Reconstruct Enabled Binding Element).**

Given a marking, \( M \), that enables a transition, \( t \in T \), a function, \( \text{ReconstructBindingElement} \), 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 7.1, is given by

\[
\text{ReconstructBindingElement} : V_{\text{subset}} \times \text{ATNames} \rightarrow \text{BE}
\]

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

\[
\text{ReconstructBindingElement}(M, \text{atn}) = (t, b)
\]

where \( (t, b) \) is a binding element from column 2 of Table 7.1 corresponding to the source marking \( M \) and an augmented transition name from column 1 of Table 7.1.

We are now ready to define our arc notation.

**Definition 7.4 (Arc Notation).**

\( a^{(M,S,R)}_{\text{atn},(s_{\text{state}},r_{\text{state}}),(ssn,rsn),(m,o,ao,mn,an,ret)} \) represents an arc, \( (M, (t, b), M') \), in \( RG^{(M,S,R)} \), with

- source marking, \( M = M^{(M,S,R)}_{(s_{\text{state}},r_{\text{state}}),(ssn,rsn),(m,o,a0,mn,an,ret)} \);
- binding element, \( (t, b) = \text{ReconstructBindingElement}(M, \text{atn}) \); 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 \( a^{(3,4)}_{\text{send}_\text{mess},(s_{\text{ready}},r_{\text{ready}}),(2,2),(1,1,0,0,0)} \) is the arc, \( (M, (t, b), M') \), where:

\[
M = M^{(3,4)}_{(s_{\text{ready}},r_{\text{ready}}),(2,2),(1,1,0,0,0)}; \\
(t, b) = (\text{send}_\text{mess}, (queue = [1], sn = 2)), \text{ and} \\
M' = M^{(3,4)}_{(\text{wait}_\text{ack},r_{\text{ready}}),(2,2),(1,1,1,0,0)}
\]
Table 7.1: A mapping from an augmented transition name, \( \text{atn} \), to its corresponding enabled binding element, given a source marking, \( M = M_{(\text{ MS}, \text{ MR})}^{(s_{\text{state}}, r_{\text{state}}), (\text{ssn}, \text{rsn}), (\text{mo}, \text{mn}, \text{an}, \text{ret})} \), that enables \( TransMap(\text{atn}) \).

<table>
<thead>
<tr>
<th>Augmented Transition Name, ( atn \in \text{AT Names} )</th>
<th>Corresponding Enabled Binding Element, ( \text{ReconstructBindingElement}(M, atn) ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{send_mess} )</td>
<td>( (\text{send_mess}, (\text{queue} = [[\text{ssn} \oplus_{\text{MS}} 1]^\text{mo} \text{ ssn}^\text{mn}], \text{sn} = \text{ssn})) ).</td>
</tr>
<tr>
<td>( \text{timeout_retrans} )</td>
<td>( (\text{timeout_retrans}, (\text{queue} = [[\text{ssn} \oplus_{\text{MS}} 1]^\text{mo} \text{ ssn}^\text{mn}], \text{sn} = \text{ssn}, \text{rc} = \text{ret})) ).</td>
</tr>
<tr>
<td>( \text{receive_mess} )</td>
<td>( (\text{receive_mess}, (\text{queue} = [[\text{ssn} \oplus_{\text{MS}} 1]^\text{mo} \text{ ssn}^\text{mn}], \text{sn} = \text{ssn} \ominus_{\text{MS}} 1, \text{rn} = \text{rsn})) ), provided ( \text{mo} &gt; 0 ); or ( (\text{receive_mess}, (\text{queue} = [[\text{ssn}^\text{mn}]-1], \text{sn} = \text{ssn}, \text{rn} = \text{rsn})) ) provided ( \text{mo} = 0 ) and ( \text{mn} &gt; 0 ).</td>
</tr>
<tr>
<td>( \text{send_ack} )</td>
<td>( (\text{send_ack}, (\text{queue} = [[\text{ssn}^\text{ao} (\text{ssn} \oplus_{\text{MS}} 1)^\text{an}], \text{rn} = \text{rsn})]) ).</td>
</tr>
<tr>
<td>( \text{receive_ack} )</td>
<td>( (\text{receive_ack}, (\text{queue} = [[\text{ssn} \oplus_{\text{MS}} 1]^\text{mo} \text{ ssn}^\text{mn}], \text{sn} = \text{ssn}, \text{rn} = \text{ssn} \ominus_{\text{MS}} 1, \text{rc} = \text{ret})) ).</td>
</tr>
<tr>
<td>( \text{receive_dup_ack} )</td>
<td>( (\text{receive_dup_ack}, (\text{queue} = [[\text{ssn}^\text{ao} \ominus_{\text{MS}} 1]^\text{an}], \text{sn} = \text{ssn}, \text{rn} = \text{ssn})) ).</td>
</tr>
<tr>
<td>( \text{mess_loss_old} )</td>
<td>( (\text{mess_loss}, (\text{queue} = [[\text{ssn} \oplus_{\text{MS}} 1]^\text{mo} \text{ ssn}^\text{mn}], \text{sn} = \text{ssn} \ominus_{\text{MS}} 1)) ), provided ( \text{mo} &gt; 0 ).</td>
</tr>
<tr>
<td>( \text{mess_loss_new} )</td>
<td>( (\text{mess_loss}, (\text{queue} = [[\text{ssn} \oplus_{\text{MS}} 1]^\text{mo} \text{ ssn}^\text{mn}], \text{sn} = \text{ssn})) ), provided ( \text{mn} &gt; 0 ).</td>
</tr>
<tr>
<td>( \text{ack_loss_old} )</td>
<td>( (\text{ack_loss}, (\text{queue} = [[\text{ssn}^\text{ao} (\text{ssn} \oplus_{\text{MS}} 1)^\text{an}], \text{rn} = \text{ssn})]) ), provided ( \text{ao} &gt; 0 ).</td>
</tr>
<tr>
<td>( \text{ack_loss_new} )</td>
<td>( (\text{ack_loss}, (\text{queue} = [[\text{ssn}^\text{ao} (\text{ssn} \oplus_{\text{MS}} 1)^\text{an}], \text{rn} = \text{ssn} \ominus_{\text{MS}} 1])) ), provided ( \text{an} &gt; 0 ).</td>
</tr>
</tbody>
</table>

which corresponds to the occurrence of the \( \text{send\_mess} \) transition, with binding \( (\text{queue} = [1], \text{sn} = 2) \), from marking \( M_{(s_{\text{ready}}, r_{\text{ready}}), (2,2),(1,1,0,0,0), \text{MS}, \text{MR}}^{(\text{s}_{\text{state}}, \text{r}_{\text{state}}, \text{ssn}, \text{rsn})} \), 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 7.1 and 7.4 is more expressive than required for the representation of all markings and arcs in \( RG_{(\text{MS}, \text{MR})} \) because not all relationships between the variables \( s_{\text{state}}, r_{\text{state}}, \text{ssn} \) and \( \text{rsn} \), or between the variables and the parameters \( \text{MS} \) and \( \text{MR} \), have been specified.
7.3 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 \( R_G(MS,MR) \) based on sender and receiver sequence number and an informal concept of the ‘class’ of marking, as was done in [48] and Section 6.1, it is worth looking more closely at the possible combinations of sender and receiver state for the general case of \( \text{MaxRetrans} \geq 0 \).

Table 7.2 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 7.1 and 7.4. 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 \( (rsn = i \oplus_{MS} 1) \).
7.4. Parametric Marking and Arc Notation

Table 7.2: Classification of markings into Classes of markings based on the state of the sender and receiver.

<table>
<thead>
<tr>
<th>M(sender_state)</th>
<th>M(receiver_state)</th>
<th>M(send_seq_no)</th>
<th>M(recv_seq_no)</th>
<th>Class_{MS}(M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1's_ready</td>
<td>1'r_ready</td>
<td>1'sn</td>
<td>1'sn</td>
<td>1</td>
</tr>
<tr>
<td>1'wait_ack</td>
<td>1'r_ready</td>
<td>1'sn</td>
<td>1'sn</td>
<td>2a</td>
</tr>
<tr>
<td>1'wait_ack</td>
<td>1'r_ready</td>
<td>1'sn</td>
<td>1'sn ⊕_{MS} 1</td>
<td>2b</td>
</tr>
<tr>
<td>1'wait_ack</td>
<td>1'process</td>
<td>1'sn</td>
<td>1'sn</td>
<td>3a</td>
</tr>
<tr>
<td>1's_ready</td>
<td>1'process</td>
<td>1'sn</td>
<td>1'sn ⊕_{MS} 1</td>
<td>3b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

- **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 we will see in Chapter 8, 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 7.5** (Classification Function).

A function, \( Class_{MS} \), which classifies the set of reachable markings according to Table 7.2, 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 7.2.

### 7.4 Parametric Marking and Arc Notation

The classes defined in Table 7.2 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 7.6** (Shorthand Marking Notation).

A reachable marking, \( M \in V_{(MS, MR)} \), is uniquely represented by \( M_{(class, ssn), (mo, ao, mn, an, ret)} \) where the superscript contains the parameter values of the SWP CPN and the subscript contains the marking description, such that \( class = Class_{MS}(M) \), and the parameters MS and MR and the variables ssn, mo, ao, mn, an and ret are as given in Definition 7.1.
7.5. Queue Lengths

**Definition 7.7** (Shorthand Arc Notation).

An arc, \((M, (t, b), M') \in A_{(MS, MR)}\), is represented by \(a_{\text{atn},(\text{class}, \text{ssn})},(\text{mo}, \text{ao}, \text{mn}, \text{an}, \text{ret})\) where \(\text{class} = \text{Class}_{MS}(M)\), and the parameters \(MS\) and \(MR\) and the variables \(\text{atn}, \text{ssn}, \text{mo}, \text{ao}, \text{mn}, \text{an}\) and \(\text{ret}\) are as given in Definition 7.4.

**Definition 7.8** (Shorthand Sets of Markings).

1. \(V_{(\text{class}, \text{ssn})}^{(MS, MR)} = \{M \in V_{(MS, MR)} \mid \text{Class}_{MS}(M) = \text{class}, M(\text{send_seq_no}) = 1'\text{ssn}\}\) represents the set of markings in which the sender sequence number is given by \(\text{ssn}\), and the sender and receiver states and receiver sequence number are given by the class as specified in Table 7.2.

2. \(V_{\text{ssn}}^{(MS, MR)} = \{M \in V_{(MS, MR)} \mid M(\text{send_seq_no}) = 1'\text{ssn}\}\) represents the set of markings in which the sender sequence number is given by \(\text{ssn}\).

**Definition 7.9** (Shorthand Sets of Arcs).

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

2. \(A_{\text{ssn}}^{(MS, MR)} = \{(M, (t, b), M') \in A_{(MS, MR)} \mid M(\text{send_seq_no}) = 1'\text{ssn}\}\) represents the set of arcs with source nodes in \(V_{\text{ssn}}^{(MS, MR)}\).

**7.5 Queue Lengths**

A function for obtaining the length of the list on place \(\text{mess\_channel}\) (the number of messages in the message channel) based on the marking notation from Definition 7.6 is defined below. An analogous function is defined for the acknowledgement channel.

**Proposition 7.1.** For a reachable marking, \(M_{(\text{class}, \text{ssn}), (\text{mo}, \text{ao}, \text{mn}, \text{an}, \text{ret})}^{(MS, MR)} \in V_{(MS, MR)}\), the number of messages in the message channel is given by the sum of the number of old messages and current messages, i.e.:

\[
|f_{c}(M(\text{mess\_channel}))| = \text{mo} + \text{mn};
\]

and the number of acknowledgements in the acknowledgement channel is given by the sum of the number of old acknowledgements and current acknowledgements, i.e.:

\[
|f_{c}(M(\text{ack\_channel}))| = \text{ao} + \text{an}.
\]

**Proof.** From Definition 7.6, the marking of \(\text{mess\_channel}\) is given by:

\[
M(\text{mess\_channel}) = 1'[(\text{ssn} \ominus_{MS} 1)^{\text{mo}} \text{ssn}^{\text{mn}}]
\]
The value corresponding to this singleton multiset is a list which contains \( mo \) instances of a message with sequence number \( ssn \ominus_{MS} 1 \) and \( mn \) instances of a message with sequence number \( ssn \). Hence \( |f_c(M(mess\_channel))| = mo + mn \). Similarly, from Definition 7.6, the marking of \( ack\_channel \) is given by:

\[
M(ack\_channel) = 1^[ssn_{ao} \ (ssn \ominus_{MS} 1)^{an}]
\]

Following the same procedure as for \( mess\_channel \), the total number of instances of an acknowledgement with sequence number \( ssn \) is given by \( ao \) and the total number of instances of an acknowledgement with sequence number \( ssn \ominus_{MS} 1 \) is given by \( an \). Hence \( |f_c(M(ack\_channel))| = ao + an \), and the proposition is proved.

### 7.6 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 \( M(mess\_channel) = 1^[1, 1, 2] \) and \( M(ack\_channel) = 1^[2] \). 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 7.10 (Downward-Closed Sets of Markings).**

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

\[
DownwardClosed(M^{(MS, MR)}(class, ssn), (mo, ao, mn, an, ret)) = \{M^{(MS, MR)}(class, ssn), (mo', ao', mn', an', ret) | 0 \leq mo' \leq mo, 0 \leq ao' \leq ao, 0 \leq mn' \leq mn, 0 \leq an' \leq an\}
\]

The function \( DownwardClosed \) can be extended to sets of markings:

**Definition 7.11 (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 \( DC : 2^V \to 2^V \)
7.7. Concluding Remarks

where:

\[ DC(V_a) = \bigcup_{M \in V_a} DownwardClosed(M) \]
\[ = \{ M^{(MS,MR)}_{(class,ssn),(mo,ao,mn,an,ret)} | M^{(MS,MR)}_{(class,ssn),(mo,ao,mn,an,ret)} \in V_a, 0 \leq mo' \leq mo, \]
\[ 0 \leq ao' \leq ao, 0 \leq mn' \leq mn, 0 \leq an' \leq an \} \]

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, Lemmas 6.8 and 6.9 allow \( DownwardClosed(M) \) to be calculated for \( M \in V_{(MS,MR)} \) by firing the \texttt{messLoss} and \texttt{ackLoss} transitions repeatedly and in every possible order until both the message and acknowledgement channels are empty.

7.7 Concluding Remarks

This chapter defines a shorthand notation that can be used to represent markings and arcs, and to define sets of markings and arcs, in a parametric way. This notation is used in the next chapter to define algebraic expressions that represent the parametric reachability graph of our parameterised SWP CPN model, and to prove that these expressions are correct.
Chapter 8

The Parametric SWP Reachability Graph

In this chapter the markings and arcs of $RG_{(MS,MR)}$ are specified using the notation from Definitions 7.6, 7.7, 7.8 and 7.9 by specifying allowable ranges of the five variables, $mo, ao, mn, an,$ and $ret$. These variables can only take non-negative values, i.e. $mo, ao, mn, an, ret \in \mathbb{N}$. The expressions for the markings and arcs of $RG_{(MS,MR)}$ are then proved correct.

The work presented in this chapter is a revised version of the work that appears in brief in [50] and in full in [51]. The structure of this chapter is as follows. The markings and arcs of our parametric reachability graph are defined in Sections 8.1 and 8.2, respectively. Section 8.3 presents the algebraic expression for the parametric reachability graph of our SWP CPN model as a theorem. This theorem is proved in Sections 8.4 and 8.5. Section 8.4 proves that the markings identified in Section 8.1 are all reachable from the initial marking. Section 8.5 proves that this set of markings is exactly the set of reachable markings, and that the arcs captured in Section 8.2 are exactly the set of arcs of the parametric reachability graph.

8.1 Sets of Markings

All of the markings of $RG_{(MS,MR)}$ are described in Table 8.1, 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, $V_{(3a,i)} = V_{(4,i)} = \emptyset$ when $MR = 0$. We can define the set of all states with the same sender sequence number:

Definition 8.1 (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 $i$,
8.2 Sets of Arcs

Table 8.1: \( V_{i}^{(MS,MR)} = \bigcup_{\text{class} \in \text{Class}} V_{(\text{class},i)}^{(MS,MR)} \), for \( 0 \leq i \leq MS \) and \( \text{Class} = \{1, 2a, 2b, 3a, 3b, 4\} \).

<table>
<thead>
<tr>
<th>Name</th>
<th>Set Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{(1,i)}^{(MS,MR)} )</td>
<td>{ ( M_{(1,i), (mo,ao,0,0,0)} ) \mid 0 \leq mo + ao \leq MR }</td>
</tr>
<tr>
<td>( V_{(2a,i)}^{(MS,MR)} )</td>
<td>{ ( M_{(2a,i), (mo,ao,mm,0,ret)} ) \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR }</td>
</tr>
<tr>
<td>( V_{(2b,i)}^{(MS,MR)} )</td>
<td>{ ( M_{(2b,i), (ao,an,an,ret)} ) \mid 0 \leq ao \leq MR, 0 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1, 0 \leq ret \leq MR }</td>
</tr>
<tr>
<td>( V_{(3a,i)}^{(MS,MR)} )</td>
<td>{ ( M_{(3a,i), (mo,ao,mm,0,ret)} ) \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR }</td>
</tr>
<tr>
<td>( V_{(3b,i)}^{(MS,MR)} )</td>
<td>{ ( M_{(3b,i), (ao,an,an,ret)} ) \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR }</td>
</tr>
<tr>
<td>( V_{(4,i)}^{(MS,MR)} )</td>
<td>{ ( M_{(4,i), (mo,ao,0,0,0)} ) \mid 0 \leq mo + ao \leq MR - 1 }</td>
</tr>
</tbody>
</table>


\( i \in \{0, 1, ..., MS\} \), where:

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

8.2 Sets of Arcs

All of the arcs of \( RG_{(MS,MR)} \) are described in Tables 8.2 to 8.7 by evaluating each of these tables for every \( i, 0 \leq i \leq MS \). There is one table of arcs per row of Table 8.1, 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 8.2** (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 8.1. 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 7.7 in Section 7.4. 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 from Definition 2.12. 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, \(A^{(MS,MR)}_{(1,i)}\), shown in Table 8.2 corresponds to all arcs with source nodes in \(V^{(MS,MR)}_{(1,i)}\), and so the values of \(mn, an\) and \(ret\) are equal to zero for all arcs in \(A^{(MS,MR)}_{(1,i)}\) due to the restrictions imposed in the definition of \(V^{(MS,MR)}_{(1,i)}\).

8.3 The Parametric Reachability Graph

We now state the theorem for our parametric reachability graph over both parameters and prove its correctness.

**Theorem 8.1.** For \(MS \in \mathbb{N}^+\) and \(MR \in \mathbb{N}\), \(RG^{(MS,MR)} = (V^{(MS,MR)}, A^{(MS,MR)})\) where

\[
V^{(MS,MR)} = \bigcup_{0 \leq i \leq MS} V^{(MS,MR)}_{i} \quad \text{and} \quad A^{(MS,MR)} = \bigcup_{0 \leq i \leq MS} A^{(MS,MR)}_{i}
\]

and all nodes and arcs are defined in Tables 8.1 to 8.7.

**Proof.** The proof is in two parts: firstly, that all states in \(V^{(MS,MR)}_{i}\) are reachable from the initial state; and secondly that there are no additional markings that can be reached from the initial marking, i.e. that \(V^{(MS,MR)}_{i}\) defines the complete set of markings. In proving the second part, we will demonstrate that every arc from every state in \(V^{(MS,MR)}_{i}\) is captured exactly by \(A^{(MS,MR)}_{i}\). The first part is proved in Section 8.4, primarily by Lemma 8.1 (along with Lemmas 8.2 and 8.3) which is named the spanning lemma, while the second part is proved in Section 8.5, by Lemma 8.4 which is named the successor lemma.

8.4 A Spanning of all Markings

To demonstrate that all states defined by \(V^{(MS,MR)}_{i}\) are reachable from the initial state, we explore a span of the reachability graph. In this part of the proof we are not concerned with all possible changes of state (i.e. all possible arcs), but only that all states in \(V^{(MS,MR)}_{i}\) are reachable from the initial state.

Recall that for each particular value \(i\) of the sender sequence number, \(0 \leq i \leq MS\), the states in \(V^{(MS,MR)}_{i}\) are split into 6 disjoint sets based on the class defined in Table 7.2. The initial marking \(M^{(MS,MR)}_{(1,0),(0,0,0,0,0)}\) belongs to the set \(V^{(MS,MR)}_{(1,0)}\) where both the sender and receiver sequence numbers equal 0 (i.e. the initial marking); or where operation of the protocol has proceeded to the point where sequence numbers
Table 8.2: The set of arcs, $A^{(MS,MR)}_{(1,i)}$, with source markings in $V^{(MS,MR)}_{(1,i)}$.  

<table>
<thead>
<tr>
<th>Row</th>
<th>Arcs</th>
<th>Binding Element</th>
<th>Destination Marking</th>
<th>New Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(MS,MR)_{\text{send/mess}}(1,i),(m,ao,0,0,0)$</td>
<td>$\text{send/mess}&lt;\text{queue} = [(i \land MS 1)^{m_0}, sn=i]$</td>
<td>$M^{(MS,MR)}_{(2a,i),(m,ao,1.0,0,0)}$</td>
<td>none</td>
</tr>
<tr>
<td>2</td>
<td>$(MS,MR)_{\text{mess/loss}}(1,i),(m,ao,0,0,0)$</td>
<td>$\text{mess/loss}&lt;\text{queue} = [(i \land MS 1)^{m_0}, sn=i \land MS 1]$</td>
<td>$M^{(MS,MR)}_{(1,i),(m-1,ao,0,0,0)}$</td>
<td>$m_0 \geq 1$</td>
</tr>
<tr>
<td>3</td>
<td>$(MS,MR)_{\text{receive/mess}}(1,i),(m,ao,0,0,0)$</td>
<td>$\text{receive/mess}&lt;\text{queue} = [(i \land MS 1)^{m_0-1}, sn=i \land MS 1, r=i]$</td>
<td>$M^{(MS,MR)}_{(4,i),(m-1,ao,0,0,0)}$</td>
<td>$m_0 \geq 1$</td>
</tr>
<tr>
<td>4</td>
<td>$(MS,MR)_{\text{ack/loss}}(1,i),(m,ao,0,0,0)$</td>
<td>$\text{ack/loss}&lt;\text{queue} = [i^{m_0}, r=i]$</td>
<td>$M^{(MS,MR)}_{(1,i),(m,ao-1,0,0,0)}$</td>
<td>$ao \geq 1$</td>
</tr>
<tr>
<td>5</td>
<td>$(MS,MR)_{\text{receive_dup_ack}}(1,i),(m,ao,0,0,0)$</td>
<td>$\text{receive_dup_ack}&lt;\text{queue} = [i^{m_0-1}, sn=i, r=i]$</td>
<td>$M^{(MS,MR)}_{(1,i),(m,ao-1,0,0,0)}$</td>
<td>$ao \geq 1$</td>
</tr>
</tbody>
</table>

Table 8.3: The set of arcs, $A^{(MS,MR)}_{(2a,i)}$, with source markings in $V^{(MS,MR)}_{(2a,i)}$.  

<table>
<thead>
<tr>
<th>Row</th>
<th>Arcs</th>
<th>Binding Element</th>
<th>Destination Marking</th>
<th>New Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(MS,MR)_{\text{timeout_retrans}}(2a,i),(m,ao,mn,0,ret)$</td>
<td>$\text{timeout_retrans}&lt;\text{queue} = [(i \land MS 1)^{m_0} i^{mn}], sn=i, rc=ret]$</td>
<td>$M^{(MS,MR)}_{(2a,i),(m,ao,mn+1,0,ret+1)}$</td>
<td>$ret &lt; MR$</td>
</tr>
<tr>
<td>2</td>
<td>$(MS,MR)_{\text{mess/loss_did}}(2a,i),(m,ao,mn,0,ret)$</td>
<td>$\text{mess/loss}&lt;\text{queue} = [(i \land MS 1)^{m_0} i^{mn}], sn=i \land MS 1]$</td>
<td>$M^{(MS,MR)}_{(2a,i),(m-1,ao,mn,0,ret)}$</td>
<td>$m_0 \geq 1$</td>
</tr>
<tr>
<td>3</td>
<td>$(MS,MR)_{\text{mess/loss_new}}(2a,i),(m,ao,mn,0,ret)$</td>
<td>$\text{mess/loss}&lt;\text{queue} = [(i \land MS 1)^{m_0} i^{mn}], sn=i$</td>
<td>$M^{(MS,MR)}_{(2a,i),(m,ao,mn+1,ret)}$</td>
<td>$mn \geq 1$</td>
</tr>
<tr>
<td>4</td>
<td>$(MS,MR)_{\text{receive/mess}}(2a,i),(m,ao,mn,0,ret)$</td>
<td>$\text{receive/mess}&lt;\text{queue} = [(i \land MS 1)^{m_0-1} i^{mn}], sn=i \land MS 1, r=i]$</td>
<td>$M^{(MS,MR)}_{(2a,i),(m-1,ao,mn-1,ret,0)}$</td>
<td>$m_0 \geq 1$</td>
</tr>
<tr>
<td>5</td>
<td>$(MS,MR)_{\text{receive_mess}}(2a,i),(m,ao,mn,0,ret)$</td>
<td>$\text{receive_mess}&lt;\text{queue} = [i^{mn-1}, sn=i, r=i]$</td>
<td>$M^{(MS,MR)}_{(2a,i),(m,ao,mn-1,ret)}$</td>
<td>$mn \geq 1$</td>
</tr>
<tr>
<td>6</td>
<td>$(MS,MR)_{\text{ack/loss_did}}(2a,i),(m,ao,mn,0,ret)$</td>
<td>$\text{ack/loss}&lt;\text{queue} = [i^{mn}], r=i]$</td>
<td>$M^{(MS,MR)}_{(2a,i),(m,ao-1,mn,0,ret)}$</td>
<td>$ao \geq 1$</td>
</tr>
<tr>
<td>7</td>
<td>$(MS,MR)_{\text{receive_dup_ack}}(2a,i),(m,ao,mn,0,ret)$</td>
<td>$\text{receive_dup_ack}&lt;\text{queue} = [i^{mn-1}, sn=i, r=i]$</td>
<td>$M^{(MS,MR)}_{(2a,i),(m,ao-1,mn,0,ret)}$</td>
<td>$ao \geq 1$</td>
</tr>
</tbody>
</table>
Table 8.4: The set of arcs, $A_{(2b, i)}^{(MS, MR)}$, with source markings in $V_{(2b, i)}^{(MS, MR)}$.

<table>
<thead>
<tr>
<th>Row</th>
<th>Arcs</th>
<th>Binding Element</th>
<th>Destination Marking</th>
<th>New Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a_{(MS, MR)}^{\text{timeout_retrans}}(2b, i), (0, ao, mn, an, ret)$</td>
<td>$\text{timeout_retrans}&lt;\text{queue} = [i^{mn}], \text{sn}=i, \text{rc}=\text{ret}&gt;$</td>
<td>$A_{(2b, i)}^{(MS, MR)}(0, ao, mn+1, an, ret+1)$</td>
<td>$\text{ret} &lt; \text{MR}$</td>
</tr>
<tr>
<td>2</td>
<td>$a_{(MS, MR)}^{\text{mess_loss}}(2b, i), (0, ao, mn, an, ret)$</td>
<td>$\text{mess_loss}&lt;\text{queue} = [i^{mn}], \text{sn}=i&gt;$</td>
<td>$M_{(2b, i)}^{(MS, MR)}(0, ao, mn-1, an, ret)$</td>
<td>$mn \geq 1$</td>
</tr>
<tr>
<td>3</td>
<td>$a_{(MS, MR)}^{\text{receive_msg}}(2b, i), (0, ao, mn, an, ret)$</td>
<td>$\text{receive_msg}&lt;\text{queue} = [i^{mn}], \text{sn}=i, m=i \oplus MS 1&gt;$</td>
<td>$M_{(2b, i)}^{(MS, MR)}(2b, i), (0, ao, mn, an, ret)$</td>
<td>$mn \geq 1$</td>
</tr>
<tr>
<td>4</td>
<td>$a_{(MS, MR)}^{\text{ack_loss}}(2b, i), (0, ao, mn, an, ret)$</td>
<td>$\text{ack_loss}&lt;\text{queue} = [i^{mn} (i \oplus MS 1)^{mn}], \text{sn}=i, \text{mn}=i \oplus MS 1&gt;$</td>
<td>$M_{(2b, i)}^{(MS, MR)}(2b, i), (0, ao, mn, an, ret)$</td>
<td>$\text{ao} \geq 1$</td>
</tr>
<tr>
<td>5</td>
<td>$a_{(MS, MR)}^{\text{send_ack}}(2b, i), (0, ao, mn, an, ret)$</td>
<td>$\text{send_ack}&lt;\text{queue} = [i^{mn}], \text{sn}=i, \text{rc}=\text{ret}&gt;, m=i \oplus MS 1, m=\text{ret}$</td>
<td>$M_{(2b, i)}^{(MS, MR)}(1, i, a;i), (mn, an, 1, 0, 0)$</td>
<td>$\text{an} \geq 1$</td>
</tr>
</tbody>
</table>

Table 8.5: The set of arcs, $A_{(3a, i)}^{(MS, MR)}$, with source markings in $V_{(3a, i)}^{(MS, MR)}$, for $MR > 0$.

<table>
<thead>
<tr>
<th>Row</th>
<th>Arcs</th>
<th>Binding Element</th>
<th>Destination Marking</th>
<th>New Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a_{(MS, MR)}^{\text{timeout_retrans}}(3a, i), (mo, ao, mn, 0, ret)$</td>
<td>$\text{timeout_retrans}&lt;\text{queue} = [(i \oplus MS 1)^{mn}], \text{sn}=i, \text{rc}=\text{ret}&gt;$</td>
<td>$M_{(3a, i)}^{(MS, MR)}(mo, ao, mn, 1, 0, ret+1)$</td>
<td>$\text{ret} &lt; \text{MR}$</td>
</tr>
<tr>
<td>2</td>
<td>$a_{(MS, MR)}^{\text{mess_loss}}(3a, i), (mo, ao, mn, 0, ret)$</td>
<td>$\text{mess_loss}&lt;\text{queue} = [(i \oplus MS 1)^{mn}], \text{sn}=i, m=i \oplus MS 1&gt;$</td>
<td>$M_{(3a, i)}^{(MS, MR)}(mo, ao, mn-1, 0, ret)$</td>
<td>$mn \geq 1$</td>
</tr>
<tr>
<td>3</td>
<td>$a_{(MS, MR)}^{\text{mess_loss}}(3a, i), (mo, ao, mn, 0, ret)$</td>
<td>$\text{mess_loss}&lt;\text{queue} = [(i \oplus MS 1)^{mn}], \text{sn}=i&gt;$</td>
<td>$M_{(3a, i)}^{(MS, MR)}(mo, ao, mn-1, 0, ret)$</td>
<td>$mn \geq 1$</td>
</tr>
<tr>
<td>4</td>
<td>$a_{(MS, MR)}^{\text{ack_loss}}(3a, i), (mo, ao, mn, 0, ret)$</td>
<td>$\text{ack_loss}&lt;\text{queue} = [i^{mn}], m=i&gt;$</td>
<td>$M_{(3a, i)}^{(MS, MR)}(mo, ao, mn-1, 0, ret)$</td>
<td>$\text{ao} \geq 1$</td>
</tr>
<tr>
<td>5</td>
<td>$a_{(MS, MR)}^{\text{receive_dupack}}(3a, i), (mo, ao, mn, 0, ret)$</td>
<td>$\text{receive_dupack}&lt;\text{queue} = [i^{mn-1}], \text{sn}=i, m=i&gt;$</td>
<td>$M_{(3a, i)}^{(MS, MR)}(mo, ao-1, mn, 0, ret)$</td>
<td>$\text{ao} \geq 1$</td>
</tr>
<tr>
<td>6</td>
<td>$a_{(MS, MR)}^{\text{send_ack}}(3a, i), (mo, ao, mn, 0, ret)$</td>
<td>$\text{send_ack}&lt;\text{queue} = [i^{mn}], m=i&gt;$</td>
<td>$M_{(3a, i)}^{(MS, MR)}(mo, ao-1, mn, 0, ret)$</td>
<td>none</td>
</tr>
</tbody>
</table>
Table 8.6: The set of arcs, $A^{(MS,MR)}_{(3b,4)}$, with source markings in $V^{(MS,MR)}_{(3b,4)}$.

<table>
<thead>
<tr>
<th>Row</th>
<th>Arcs</th>
<th>Binding Element</th>
<th>Destination Marking</th>
<th>New Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\delta^{(MS,MR)}_{\text{timeout_retrans}}(3b,i),(0,ao,mm,an,ret)$</td>
<td>timeout_retrans</td>
<td>$&lt;\text{queue} = [i^{mn}], \text{sn}=i, \text{rc}=\text{ret}&gt;$</td>
<td>$M^{(MS,MR)}_{(3b,i),(0,ao,mm+1,an,ret+1)}$</td>
</tr>
<tr>
<td>2</td>
<td>$\delta^{(MS,MR)}_{\text{mess_loss}}(3b,i),(0,ao,mm,an,ret)$</td>
<td>mess_loss</td>
<td>$&lt;\text{queue} = [i^{mn}], \text{sn}=i&gt;$</td>
<td>$M^{(MS,MR)}_{(3b,i),(0,ao,mm-1,an,ret)}$</td>
</tr>
<tr>
<td>3</td>
<td>$\delta^{(MS,MR)}_{\text{ack_loss}}(3b,i),(0,ao,mm,an,ret)$</td>
<td>ack_loss</td>
<td>$&lt;\text{queue} = [i^{ao}(i+MS1)^{an}], \text{rn}=i&gt;$</td>
<td>$M^{(MS,MR)}_{(3b,i),(0,ao-1,mm,an,ret)}$</td>
</tr>
<tr>
<td>4</td>
<td>$\delta^{(MS,MR)}_{\text{ack_loss}}(3b,i),(0,ao,mm,an,ret)$</td>
<td>ack</td>
<td>$&lt;\text{queue} = [i^{an}(i+MS1)^{mn}], \text{rn}=i+MS1&gt;$</td>
<td>$M^{(MS,MR)}_{(3b,i),(0,ao,mm-1,an,ret)}$</td>
</tr>
<tr>
<td>5</td>
<td>$\delta^{(MS,MR)}_{\text{receive_dup_ack}}(3b,i),(0,ao,mm,an,ret)$</td>
<td>receive_dup_ack</td>
<td>$&lt;\text{queue} = [i^{ao-1}(i+MS1)^{an}], \text{sn}=i, \text{rn}=i&gt;$</td>
<td>$M^{(MS,MR)}_{(3b,i),(0,ao-1,mm,an,ret)}$</td>
</tr>
<tr>
<td>6</td>
<td>$\delta^{(MS,MR)}_{\text{receive_ack}}(3b,i),(0,ao,mm,an,ret)$</td>
<td>receive_ack</td>
<td>$&lt;\text{queue} = [(i+MS1)^{an-1}], \text{sn}=i, \text{rn}=i+MS1, \text{rc}=\text{ret}&gt;$</td>
<td>$M^{(MS,MR)}_{(4i\oplus MS1),(mm,an-1,0,0,0)}$</td>
</tr>
<tr>
<td>7</td>
<td>$\delta^{(MS,MR)}_{\text{send_ack}}(3b,i),(0,ao,mm,an,ret)$</td>
<td>send_ack</td>
<td>$&lt;\text{queue} = [i^{ao}(i+MS1)^{an}], \text{rn}=i+MS1&gt;$</td>
<td>$M^{(MS,MR)}_{(2b,i),(0,ao,mm+1,an,ret)}$</td>
</tr>
</tbody>
</table>

Table 8.7: The set of arcs, $A^{(MS,MR)}_{(4,i)}$, with source markings in $V^{(MS,MR)}_{(4,i)}$, for $MR > 0$.

<table>
<thead>
<tr>
<th>Row</th>
<th>Arcs</th>
<th>Binding Element</th>
<th>Destination Marking</th>
<th>New Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\delta^{(MS,MR)}_{\text{send_msg}}(4,i),(mo,ao,0,0,0)$</td>
<td>send_msg</td>
<td>$&lt;\text{queue} = [(i+MS1)^{mn}], \text{sn}=i&gt;$</td>
<td>$M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)}$</td>
</tr>
<tr>
<td>2</td>
<td>$\delta^{(MS,MR)}_{\text{mess_loss}}(4,i),(mo,ao,0,0,0)$</td>
<td>mess_loss</td>
<td>$&lt;\text{queue} = [(i+MS1)^{mn}], \text{sn}=i+MS1&gt;$</td>
<td>$M^{(MS,MR)}_{(4,i),(mo-1,ao,0,0,0)}$</td>
</tr>
<tr>
<td>3</td>
<td>$\delta^{(MS,MR)}_{\text{ack_loss}}(4,i),(mo,ao,0,0,0)$</td>
<td>ack_loss</td>
<td>$&lt;\text{queue} = [i^{an}], \text{rn}=i&gt;$</td>
<td>$M^{(MS,MR)}_{(4,i),(mo,ao-1,0,0,0)}$</td>
</tr>
<tr>
<td>4</td>
<td>$\delta^{(MS,MR)}_{\text{receive_dup_ack}}(4,i),(mo,ao,0,0,0)$</td>
<td>receive_dup_ack</td>
<td>$&lt;\text{queue} = [i^{ao-1}], \text{sn}=i, \text{rn}=i&gt;$</td>
<td>$M^{(MS,MR)}_{(4,i),(mo,ao-1,0,0,0)}$</td>
</tr>
<tr>
<td>5</td>
<td>$\delta^{(MS,MR)}_{\text{send_ack}}(4,i),(mo,ao,0,0,0)$</td>
<td>send_ack</td>
<td>$&lt;\text{queue} = [i^{ao}], \text{rn}=i&gt;$</td>
<td>$M^{(MS,MR)}_{(4,i),(mo,ao+1,0,0,0)}$</td>
</tr>
</tbody>
</table>
8.4. A Spanning of all Markings

have wrapped, the sender and receiver sequence numbers have returned to 0, and all duplicate messages and acknowledgements have been either lost or received and processed accordingly (i.e. the channels are empty).

This conceptual complication doesn’t exist with any other marking. Consequently, the markings in \( V_{\text{MS};\text{MR}}(1;0) \) in which duplicate messages (with sequence number \( MS \)) or duplicate acknowledgements (with sequence number 0) exist cannot be reached from the initial marking until after sequence numbers have wrapped.

As an aside, this conceptual complication explains why we chose to include the empty service in the Stop-and-Wait Service Language in Section 5.2.1. If the empty service were excluded, then the initial marking would need to be differentiated from a ‘new’ marking in which sequence numbers have wrapped back to 0, both sender and receiver are in their ready states, and both channels are empty.

Returning to the proof of Theorem 8.1, we now give an intuitive feel for how the proof is structured. To avoid the complications that arise from the fact that not all markings in \( V_{\text{MS};\text{MR}}(1;0) \) are reachable until after sequence numbers have wrapped, we show that all the markings in \( V_{\text{MS};\text{MR}}(1;0) \cup \{ M_{\text{MS};\text{MR}}(1,i;\text{MR},0,0,0,0) \} \) are reachable from \( M_{\text{MS};\text{MR}}(1,i;\text{MR},0,0,0,0) \), where including \( M_{\text{MS};\text{MR}}(1,i;\text{MR},0,0,0,0) \) in this set provides a link from one set to the next, i.e. from the set with sender sequence number \( i \) to the set with sequence number \( i \oplus MS 1 \). This link allows these sets to be strung together for all \( i \in \{0, 1, ..., MS\} \), and we show that for all \( i, j \in \{0, 1, ..., MS\} \), \( M_{\text{MS};\text{MR}}(1,i;\text{MR},0,0,0,0) \in [M_{\text{MS};\text{MR}}(1,j;\text{MR},0,0,0,0)] \).

As a final step, however, we must demonstrate that the initial marking can reach one of the markings in \( \{ M_{\text{MS};\text{MR}}(1,i;\text{MR},0,0,0,0) \mid 0 \leq i \leq MS \} \) and hence all markings in \( V_{\text{MS};\text{MR}} \).

This approach is illustrated by the components with solid outlines in Fig. 8.1. The small circles represent markings, the arrows represent arcs, and the two large curved outlines represent the sets of
marks, $\overline{V}_i^{(MS,MR)}$ and $\overline{V}_{i\in MS}^{(MS,MR)}$. The two markings, $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$ and $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$, are shown at the top of their respective sets of markings. From the marking $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$, we shall find a spanning of the markings in $\overline{V}_i^{(MS,MR)}$, including an arc that leads from some marking in $\overline{V}_i^{(MS,MR)}$ to the marking $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$. The components with the dashed outline show that, just as a spanning of all markings in $\overline{V}_i^{(MS,MR)}$ starting from $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$ can be discovered, so can a spanning of all markings in $\overline{V}_{i\in MS}^{(MS,MR)}$ starting from $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$, and so on. The final step of discovering a path from the initial marking to a marking in $\{M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)} | 0 \leq i \leq MS\}$ is not shown in Fig. 8.1.

Lemma 8.1. Spanning Lemma. For a given $MS \in \mathbb{N}^+$ and $MR \in \mathbb{N}$, for $0 \leq i \leq MS$, all markings in $\overline{V}_i^{(MS,MR)} \cup \{M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}\}$ are reachable from $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$.

Proof. The following 6 sublemmas describe one possible spanning of the markings within each of the 6 subsets of markings $\overline{V}_i^{(MS,MR)}$, $V_{2a(i)}^{(MS,MR)}$, $V_{2b(i)}^{(MS,MR)}$, $V_{3a(i)}^{(MS,MR)}$, $V_{3b(i)}^{(MS,MR)}$ and $V_{4(i)}^{(MS,MR)}$, of $\overline{V}_i^{(MS,MR)}$. The sublemmas are all very similar in structure. In cases where retransmissions are possible (classes 2a, 3a, 3b and 2b in Sublemmas 8.1.2 to 8.1.5) we begin by firing timeout_retrans as many times as we can. From the resultant set of markings (a single marking in the case of class 1 and 4, in Sublemmas 8.1.1 and 8.1.6) we then fire repetitions of receive_mess, send_ack (Sublemmas 8.1.1, 8.1.2 and 8.1.5) or send_ack, receive_mess (Sublemmas 8.1.3, 8.1.4 and 8.1.6). For each sublemma, the downward-closed set of the resulting set of markings spans all markings in the respective class’s subset of $\overline{V}_i^{(MS,MR)}$. Reachability of markings between these subsets is demonstrated and is illustrated as we successively prove each sublemma.

Sublemma 8.1.1. All markings in $\overline{V}_{1(i)}^{(MS,MR)}$ are reachable from $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$.

Proof. $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$ describes a marking where both sender and receiver are in their ready states with ssn and rsn $= i$ ($0 \leq i \leq MS$), the sender’s retransmission counter is 0, $MR$ retransmitted messages with sequence number $i \in MS$ 1 are in the message channel and the acknowledgement channel is empty. $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$ represents the marking $M$ where:

\[
\begin{align*}
M(\text{sender\_state}) &= 1^i_s\text{\_ready} & M(\text{send\_seq\_no}) &= 1^i_i \\
M(\text{receiver\_state}) &= 1^i_r\text{\_ready} & M(\text{recv\_seq\_no}) &= 1^i_i \\
M(\text{mess\_channel}) &= 1^i[(i \in MS, 1)^{MR}] & M(\text{ack\_channel}) &= 1^i[\]
M(\text{retrans\_counter}) &= 1^i 0
\end{align*}
\]

This marking enables the receive_mess transition with binding $<sn = i \in MS, 1, rn = i, queue = [(i \in MS, 1)^{MR-1}>$ where $(i \in MS, 1)^{MR-1}$ indicates $MR - 1$ repetitions of the message with sequence number $i \in MS, 1$. From the transition rule and Lemmas 6.4 and 6.5, each of the messages with sequence
8.4. A Spanning of all Markings

\[ V_{\text{span}}^{\text{class} 1} = \{ M_{(1,i),(MR-x,x,0,0,0)}^{(MS,MR)} \mid 0 \leq x \leq MR \} \]  

Technically, Equation (8.1) defines a family of sets of markings, parameterised by \( MS \) and \( MR \), for each value of \( i \), \( 0 \leq i \leq MS \). These details are omitted from the name of the set, \( V_{\text{span} \text{class} 1} \), for notational simplicity and also to avoid confusion with the notation for sets of markings from Definition 7.8. From context the omitted details (\( MS \), \( MR \) and \( i \)) should be evident. This same notational simplification is also used in Sublemmas 8.1.2 to 8.1.6.

From Lemmas 6.8 and 6.9, each marking, \( M \in V_{\text{span} \text{class} 1} \), enables either the mess\_loss transition, the ack\_loss transition, or both. As noted in Section 7.6, for each \( M \in V_{\text{span} \text{class} 1} \), it is possible to reach every marking in the downward-closed set of \( M \) by repeated firing of the mess\_loss and ack\_loss transitions until the message and acknowledgement channels are empty. This is illustrated in Fig. 8.3 for the mess\_loss transition and in Fig. 8.4 for both mess\_loss and ack\_loss. Again, details of the binding elements for each arc have been omitted for simplicity.

Formalising, this means that, for each value of \( x \), \( 0 \leq x \leq MR \), the marking \( M_{(1,i),(MR-x,x,0,0,0)}^{(MS,MR)} \) can reach the markings in \( \{ M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} \mid mo \leq MR - x \text{ and } ao \leq x \} \). From Equation (8.1) and
substituting Definition 7.11, the set of downward-closed markings of the set $V_{\text{span\_class1}}$ is the set:

$$V_{\text{dc\_span\_class1}} = DC(V_{\text{span\_class1}})$$

$$= \{ M^{(MS,MR)}_{(1,i),(m_0,a_0,0,0)} \mid M^{(MS,MR)}_{(1,i),(MR-x,x,0,0)} \in V_{\text{span\_class1}}, 0 \leq m_0 \leq MR - x, 0 \leq a_0 \leq x \}$$

$$= \{ M^{(MS,MR)}_{(1,i),(m_0,a_0,0,0)} \mid 0 \leq m_0 \leq MR - x, 0 \leq a_0 \leq x, 0 \leq x \leq MR \}$$

The inequalities in $V_{\text{dc\_span\_class1}}$ can be simplified by eliminating the variable $x$. Summing together the corresponding terms from $0 \leq m_0 \leq MR - x$ and $0 \leq a_0 \leq x$, we get $0 \leq m_0 + a_0 \leq MR$. Intuitively, as $x$ varies from 0 to $MR$, the maximum value $m_0$ can take moves from $MR$ to 0 and the maximum value $a_0$ can take moves from 0 to $MR$. Hence, the sum of $m_0$ and $a_0$ can never be greater than $MR - x + x = MR$. The inequality $0 \leq x \leq MR$ becomes redundant. Performing this simplification results in the expression:

$$V_{\text{dc\_span\_class1}} = \{ M^{(MS,MR)}_{(1,i),(m_0,a_0,0,0)} \mid 0 \leq m_0 + a_0 \leq MR \}$$

(8.2)

from the definition of $V^{(MS,MR)}_{(1,i)}$ in row 1 of Table 8.1. Thus the sublemma is proved.

Consider Fig. 8.5. It shows an abstract view of the spanning of all markings in $V^{(MS,MR)}_{(1,i)}$ from $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$ shown in Figs. 8.3 and 8.4. From $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)} \in V^{(MS,MR)}_{(1,i)}$ we can reach $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)}$ by the occurrence of transition send\_mess with binding $<queue = [(i \oplus MS 1)^{MR}]$, $sn = i>$ (from Lemma 6.2 and the transition rule).

The next sublemma shows that all markings in $V^{(MS,MR)}_{(2a,i)}$ are reachable from this same marking, $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)}$.

**Sublemma 8.1.2.** All markings in $V^{(MS,MR)}_{(2a,i)}$ are reachable from $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)}$.

**Proof.** From Definition 7.6, $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)}$ represents the marking $M$ where:

- $M(\text{sender\_state}) = 1^i\text{wait\_ack}$
- $M(\text{receiver\_state}) = 1^i\text{r\_ready}$
- $M(\text{mess\_channel}) = 1^i[(i \oplus MS 1)^{MR} i^1]$
- $M(\text{retrans\_counter}) = 1^0$

From Lemma 6.3 this marking enables transition timeout\_retrans with binding $<queue = [(i \oplus MS 1)^{MR} i^1]$, $sn = i, rec = 0>$ for $i$ in the range $0 \leq i \leq MS$. The occurrence of timeout\_retrans adds one to the retransmission counter and a new message to the message channel. By $MR$ successive occurrences of transition timeout\_retrans (i.e. until the retransmission counter reaches its maximum
8.4. A Spanning of all Markings

Figure 8.3: The class 1 markings from $V_i^{(MS,MR)}$, represented by the value of $mo$ and $ao$, reachable from the markings in Fig. 8.2 through repeated firing of messLoss.

Figure 8.4: The class 1 markings from $V_i^{(MS,MR)}$, represented by the value of $mo$ and $ao$, reachable from the markings in Fig. 8.2 by considering all possible interleavings of messLoss and ackLoss.
8.4. A Spanning of all Markings

Figure 8.5: An abstract view of the spanning of all markings in the set $V^{(MS,MR)}_{(1,i)}$, with an arc from $M^{(MS,MR)}_{(1,i),(MR,0,0,0)}$ to $M^{(MS,MR)}_{(2a,i),(MR,0,1,0)}$ for the future spanning of all markings in $V^{(MS,MR)}_{(2a,i)}$.

value and timeout_retrans is no longer enabled) the marking $M^{(MS,MR)}_{(2a,i),(MR,0,1,0)}$ can reach the markings in $V_{span\text{class}\,2a}$:

$$V_{span\text{class}\,2a} = \{ M^{(MS,MR)}_{(2a,i),(MR,0,0,ret+1,0,ret)} \mid 0 \leq ret \leq MR \}$$

From Lemmas 6.4 and 6.5 and the transition rule, transitions receive_mess and send_ack can be fired in succession, $MR$ number of times, from each marking in $V_{span\text{class}\,2a}$. This results in the set of markings:

$$V'_{span\text{class}\,2a} = \bigcup_{0 \leq x \leq MR} \{ M^{(MS,MR)}_{(2a,i),(MR-x,x,ret+1,0,ret)} \mid 0 \leq ret \leq MR \}$$

$$= \{ M^{(MS,MR)}_{(2a,i),(MR-x,x,ret+1,0,ret)} \mid 0 \leq x \leq MR, 0 \leq ret \leq MR \}$$

(8.3)

By firing receive_mess, send_ack $MR$ number of times, we are receiving all of the old duplicate messages and generating acknowledgements for these old messages, without affecting any of the instances of the new message in the message channel. The reason for not wanting to receive any new messages is that the reception of the first new message will cause the receiver to increment its sequence number (as the new message with sequence number $i$ is the next expected message) and so when send_ack is fired after receiving the first new message the resultant marking will be a class 2b marking, not a class 2a marking.

Finally, the downward-closed set of markings reachable from the set of markings $V'_{span\text{class}\,2a}$ can
8.4. A Spanning of all Markings

Figure 8.6: An abstract view of the spanning of all markings in $V^{(MS,MR)}_{(1,i)}$ and $V^{(MS,MR)}_{(2a,i)}$, with an arc from $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)}$ to $M^{(MS,MR)}_{(3a,i),(MR−1,0,1,0,0)}$ for the future spanning of all markings in $V^{(MS,MR)}_{(3a,i)}$.

be obtained. From Equation (8.3) and Definition 7.11 this set of markings is:

$$V_{dc, span \_class_{2a}} = DC(V'_{span \_class_{2a}})$$

$$= \{ M^{(MS,MR)}_{(2a,i),(mo,ao,mm,0,ret)} \mid M^{(MS,MR)}_{(2a,i),(MR−x,x,ret+1,0,ret)} \in V'_{span \_class_{2a}},$$

$$0 \leq mo \leq MR − x, 0 \leq ao \leq x, 0 \leq mn \leq ret + 1 \}$$

$$= \{ M^{(MS,MR)}_{(2a,i),(mo,ao,mm,0,ret)} \mid 0 \leq mo \leq MR − x, 0 \leq ao \leq x, 0 \leq mn \leq ret + 1, 0 \leq x \leq MR, 0 \leq ret \leq MR \}$$

The inequalities involving $mo$, $ao$ and $x$ can be simplified as in Sublemma 8.1.1 to give the inequality $0 \leq mo + ao \leq MR$. Hence, $V_{dc, span \_class_{2a}}$ can be expressed as:

$$V_{dc, span \_class_{2a}} = \{ M^{(MS,MR)}_{(2a,i),(mo,ao,mm,0,ret)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR \}$$

(8.4)

from the definition of $V^{(MS,MR)}_{(2a,i)}$ in row 2 of Table 8.1. Thus the sublemma is proved.

Figure 8.6 shows an abstract view of the spanning of all markings in $V^{(MS,MR)}_{(1,i)}$ and $V^{(MS,MR)}_{(2a,i)}$. For the situation where $MR > 0$, the marking $M^{(MS,MR)}_{(3a,i),(MR−1,0,1,0,0)}$ can be reached from the marking $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)} \in V^{(MS,MR)}_{(2a,i)}$ by the occurrence of transition receive \_mess with binding $<queue = [(i \ominus MS 1)^{MR−1} i^1], sn = i \ominus MS 1, rn = i>$ (from the transition rule and Lemma 6.4). This corresponds
8.4. A Spanning of all Markings

to the reception of an old duplicate message by the receiver. (When \( MR = 0 \), this particular transition will not be enabled by \( M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)} \) as there will be no old duplicate messages in the message channel.)

The next sublemma shows that when \( MR > 0 \), all markings in \( V^{(MS,MR)}_{(3a,i)} \) are reachable from this same marking, \( M^{(MS,MR)}_{(3a,i),(MR-1,0,1,0,0)} \).

**Sublemma 8.1.3.** When \( MR > 0 \), all markings in \( V^{(MS,MR)}_{(3a,i)} \) are reachable from \( M^{(MS,MR)}_{(3a,i),(MR-1,0,1,0,0)} \).

**Proof.** From Definition 7.6, \( M^{(MS,MR)}_{(3a,i),(MR,0,1,0,0)} \) represents the marking \( M \) where:

\[
\begin{align*}
M(\text{sender state}) &= 1'\text{wait,ack} \\
M(\text{receiver state}) &= 1'\text{process} \\
M(\text{mess,channel}) &= 1'[(i \odot MS 1)^{MR-1} i^1] \\
M(\text{retrans,counter}) &= 1'0
\end{align*}
\]

The proof of this sublemma proceeds in exactly the same way as the proof of Sublemma 8.1.2. From Lemma 6.3 this marking enables transition \( \text{timeout, retrans} \) with binding \( <\text{queue} = [(i \odot MS 1)^{MR-1} i^1], sn = i, rc = 0> \). By \( MR \) successive occurrences of transition \( \text{timeout, retrans} \) we can reach the markings in \( V^{\text{span, class 3a}} \):

\[
V^{\text{span, class 3a}} = \{ M^{(MS,MR)}_{(3a,i),(MR-1,0,1,0,0),ret+1,0,ret} \mid 0 \leq ret \leq MR, MR > 0 \}
\]

From Lemmas 6.5 and 6.4 and the transition rule, transitions \( \text{send,ack} \) and \( \text{receive, mess} \) can be fired in succession, \( MR \) number of times, from each marking in \( V^{\text{span, class 3a}} \). This results in the set of markings \( V^{\prime}_{\text{span, class 3a}} \):

\[
V^{\prime}_{\text{span, class 3a}} = \bigcup_{0 \leq x \leq MR-1} \{ M^{(MS,MR)}_{(3a,i),(MR-1-x,x,ret+1,0,ret)} \mid 0 \leq ret \leq MR, MR > 0 \}
\]
\[
= \{ M^{(MS,MR)}_{(3a,i),(MR-1-x,x,ret+1,0,ret)} \mid 0 \leq x \leq MR-1, 0 \leq ret \leq MR, MR > 0 \}
\]

Like in Sublemma 8.1.2, we only consider the repeated firing of \( \text{send,ack} \) and \( \text{receive, mess} \) \( MR-1 \) times to each marking in \( V^{\text{span, class 3a}} \) so that none of the instances of the new message are received by the receiver. Otherwise, we would end up in a class 3b marking, not a class 3a marking.

Finally, the downward-closed set of markings reachable from \( V^{\prime}_{\text{span, class 3a}} \) can be obtained. From
8.4. A Spanning of all Markings

Equation (8.5) and Definition 7.11 this set of markings is:

\[
V_{\text{span-class,3a}} = DC(V'_{\text{span-class,3a}})
\]
\[
= \{ M_{(3a,i),(mo,ao,mm,0,ret)}^{(MS,MR)} \mid M_{(3a,i),(MR-1-x,x,ret,1,0,ret)}^{(MS,MR)} \in V_{\text{span-class,2a}},
\]
\[
0 \leq mo \leq MR - 1 - x, 0 \leq ao \leq x, 0 \leq mn \leq ret + 1 \}
\]
\[
= \{ M_{(3a,i),(mo,ao,mm,0,ret)}^{(MS,MR)} \mid 0 \leq mo \leq MR - 1 - x, 0 \leq ao \leq x, 0 \leq mn \leq ret + 1,
\]
\[
0 \leq x \leq MR - 1, 0 \leq ret \leq MR, MR > 0 \}
\]

The inequalities involving \(mo, ao\) and \(x\) can be simplified in a similar way to the two previous sublemmas. Summing the corresponding terms from \(0 \leq mo \leq MR - 1 - x\) and \(0 \leq ao \leq x\), we get \(0 \leq mo + ao \leq MR - 1\). Intuitively, as \(x\) varies from 0 to \(MR - 1\), the maximum possible value of \(mo\) varies from \(MR - 1\) to 0, and the maximum possible value of \(ao\) varies from 0 to \(MR - 1\). Hence, the sum of \(mo\) and \(ao\) can never be greater than \(MR - 1\). The inequality \(0 \leq x \leq MR - 1\) becomes redundant. Performing this simplification results in the expression:

\[
V_{\text{span-class,3a}} = \{ M_{(3a,i),(mo,ao,mm,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq ret + 1,
\]
\[
0 \leq ret \leq MR, MR > 0 \}
\]

from the definition of \(V_{(3a,i)}^{(MS,MR)}\) in row 4 of Table 8.1. Thus the sublemma is proved.

Figure 8.7 shows an abstract view of the spanning of all markings in \(V_{(1,i)}^{(MS,MR)}\), \(V_{(2a,i)}^{(MS,MR)}\) and \(V_{(3a,i)}^{(MS,MR)}\). From \(M_{(2a,i),(0,MR,1,0,0)}^{(MS,MR)} \in V_{(2a,i)}^{(MS,MR)}\) we can reach \(M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}\) by the occurrence of transition \texttt{receive.mess} with binding \(<\text{queue} = [], \text{sn} = i, \text{rn} = i>\) (from Lemma 6.4 and the transition rule). This marking is chosen because the occurrence of \texttt{receive.mess} corresponds to the reception of the first new message, with the least number of retransmissions (0) and the most number of old acknowledgements (\(MR\)) out of all markings in \(V_{(2a,i)}^{(MS,MR)}\). Hence, we can apply the same technique as in the previous three sublemmas to show that all markings in \(V_{(3a,i)}^{(MS,MR)}\) are reachable from \(M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}\).

**Sublemma 8.1.4.** All markings in \(V_{(3a,i)}^{(MS,MR)}\) are reachable from \(M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}\).

**Proof.** From Definition 7.6, \(M_{(3b,i),(0,MR,0,0,0)}^{(MS,MR)}\) represents the marking \(M\) where:

\[
M(\text{sender.state}) = 1^1\text{wait.ack} \quad M(\text{send.seq.no}) = 1^1i
\]
\[
M(\text{receiver.state}) = 1^1\text{process} \quad M(\text{recv.seq.no}) = 1^1i \oplus_{MS} 1
\]
\[
M(\text{mess.channel}) = 1^1[] \quad M(\text{ack.channel}) = 1^1[i^{1MR}]
\]
\[
M(\text{retrans.counter}) = 1^10
\]
8.4. A Spanning of all Markings

Figure 8.7: An abstract view of the spanning of all markings in $V^{(MS,MR)}_{(1,i)}$, $V^{(MS,MR)}_{(2a,i)}$ and $V^{(MS,MR)}_{(3a,i)}$, with an arc from $M^{(MS,MR)}_{(2a,i),(0,MR,1,0,0)}$ to $M^{(MS,MR)}_{(3b,i),(0,MR,0,0,0)}$ for the future spanning of all markings in $V^{(MS,MR)}_{(3b,i)}$.

As in the previous two sublemmas, from Lemma 6.3, this marking enables transition $\text{timeout}_\text{retrans}$ with binding $<queue = [], sn = i, rc = 0>$. By $MR$ successive occurrences of transition $\text{timeout}_\text{retrans}$ we can reach the markings in $V^{(MS,MR)}_{\text{span\_class}\ 3b}$:

$$V^{(MS,MR)}_{\text{span\_class}\ 3b} = \{ M^{(MS,MR)}_{(3b,i),(0,MR,ret,0,ret)} \mid 0 \leq ret \leq MR \}$$

Similar to the proof of Sublemma 8.1.3, we can fire $\text{send\_ack}$ and $\text{receive\_mess}$ successively, $ret$ number of times from each marking in $V^{(MS,MR)}_{\text{span\_class}\ 3b}$ (by Lemmas 6.5 and 6.4 and the transition rule). This results in the set of markings $V^{(MS,MR)}_{\text{span\_class}\ 3b}$:

$$V^{(MS,MR)}_{\text{span\_class}\ 3b} = \bigcup_{0 \leq x \leq ret} \{ M^{(MS,MR)}_{(3b,i),(0,MR,ret-x,x,ret)} \mid 0 \leq ret \leq MR \}$$

$$= \{ M^{(MS,MR)}_{(3b,i),(0,MR,ret-x,x,ret)} \mid 0 \leq x \leq ret, 0 \leq ret \leq MR \}$$

(8.7)

We consider $ret$ successive occurrences of $\text{send\_ack}$ and $\text{receive\_mess}$ because there are $ret$ instances of the new message in the message channel in each marking in $V^{(MS,MR)}_{\text{span\_class}\ 3b}$. After $ret$ repetitions, all new messages will have been received and acknowledgements sent, except for the last message which will be being processed by the receiver. At this point, the message channel is empty.
Finally, the downward-closed set of markings reachable from $V'_{\text{span\_class\_3b}}$ can be obtained. From Equation (8.7) and Definition 7.11 this set of markings is:

\[
V_{\text{dc\_span\_class\_3b}} = DC(V'_{\text{span\_class\_3b}}) = \{ M^{(MS,MR)}_{(3b,i),(0,ao,an,ret)} | 0 \leq ao \leq MR, 0 \leq mn \leq ret - x, 0 \leq an \leq x \},
\]

Simplifying the three inequalities involving $mn, an$ and $x$ proceed in the same way as simplifying the inequalities involving $mo, ao$ and $x$ in the previous sublemmas, but with $mn$ and $an$ instead of $mo$ and $ao$, and with $MR$ replaced by $ret$ in this case. The three inequalities, $0 \leq mn \leq ret - x, 0 \leq an \leq x$ and $0 \leq x \leq ret$, simplify to $0 \leq mn + an \leq ret$. This simplification allows $V_{\text{dc\_span\_class\_3b}}$ to be expressed as:

\[
V_{\text{dc\_span\_class\_3b}} = \{ M^{(MS,MR)}_{(3b,i),(0,ao,an,ret)} | 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR \}
\]

from the definition of $V^{(MS,MR)}_{(3b,i)}$ in row 5 of Table 8.1. Thus the sublemma is proved.

Figure 8.8 shows an abstract view of the spanning of all markings in $V^{(MS,MR)}_{(1,i)}, V^{(MS,MR)}_{(2a,i)}, V^{(MS,MR)}_{(3a,i)}$ and $V^{(MS,MR)}_{(3b,i)}$. From $M^{(MS,MR)}_{(3b,i),(0,MR,0,0,0)} \in V^{(MS,MR)}_{(3b,i)}$ we can reach $M^{(MS,MR)}_{(2b,i),(0,MR,0,1,0)}$ by the occurrence of transition $\text{send\_ack}$ with binding $<queue = [i^{MR}], rn = i \oplus MS1>$ (from Lemma 6.5 and the transition rule). The next sublemma shows that all markings in $V^{(MS,MR)}_{(2b,i)}$ are reachable from this same marking, $M^{(MS,MR)}_{(2b,i),(0,MR,0,1,0)}$.

**Sublemma 8.1.5.** All markings in $V^{(MS,MR)}_{(2b,i)}$ are reachable from $M^{(MS,MR)}_{(2b,i),(0,MR,0,1,0)}$.

**Proof.** From Definition 7.6, $M^{(MS,MR)}_{(2b,i),(0,MR,0,1,0)}$ represents the marking $M$ where:

\[
\begin{align*}
M(\text{sender\_state}) &= 1\text{\_wait\_ack} & M(\text{send\_seq\_no}) &= 1^i \\
M(\text{receiver\_state}) &= 1\text{\_r\_ready} & M(\text{recv\_seq\_no}) &= 1^i \oplus MS 1 \\
M(\text{mess\_channel}) &= 1^i[ & M(\text{ack\_channel}) &= 1^i[i^{MR} (i \oplus MS1)] \\
M(\text{retrans\_counter}) &= 1^0
\end{align*}
\]

As has been the case for the previous three sublemmas, this marking enables $\text{timeout\_retrans}$ with binding $<queue = [], sn = i, rc = 0>$. By $MR$ successive occurrences of $\text{timeout\_retrans}$ we can
8.4. A Spanning of all Markings

Figure 8.8: An abstract view of the spanning of all markings in \( V^{(MS,MR)}_{(1,i)} \), \( V^{(MS,MR)}_{(2a,i)} \), and \( V^{(MS,MR)}_{(2b,i)} \), with an arc from \( M^{(MS,MR)}_{(3b,i),(0,MR,0,0,0)} \) to \( M^{(MS,MR)}_{(2b,i),(0,MR,0,1,0,0)} \) for the future spanning of all markings in \( V^{(MS,MR)}_{(2b,i)} \).

reach the markings:

\[
V_{span, class 2b} = \{ M^{(MS,MR)}_{(2b,i),(0,MR,ret,1,ret)} \mid 0 \leq ret \leq MR \}
\]

Like in the proofs of Sublemmas 8.1.1 and 8.1.2, these markings enable receive_mess. By Lemmas 6.4 and 6.5 and the transition rule, transitions receive_mess and send_ack can be successively
applied, \( ret \) times, to each marking in \( V_{\text{span class}2b} \). Hence, we can reach the markings in \( V'_{\text{span class}2b} \):

\[
V'_{\text{span class}2b} = \bigcup_{0 \leq x \leq ret} \{ M_{(2b,i),(0, MR, ret-x,1+x, ret)}^{(MS,MR)} \mid 0 \leq ret \leq MR \} \\
= \{ M_{(2b,i),(0, MR, ret-x,1+x, ret)}^{(MS,MR)} \mid 0 \leq x \leq ret, 0 \leq ret \leq MR \}
\]

(8.9)

As before, we only consider \( ret \) repetitions of receive\_mess and send\_ack because the message channel of each marking in \( V_{\text{span class}2b} \) contains \( ret \) new messages only.

Finally, the downward-closed set of markings reachable from the markings in \( V'_{\text{span class}2b} \) can be obtained. From Equation (8.9) and Definition 7.11 this set of markings is:

\[
V_{dc,\text{span class}2b} = DC(V'_{\text{span class}2b}) \\
= \{ M_{(2b,i),(0, ao, mn, an, ret)}^{(MS,MR)} \mid M_{(2b,i),(0, MR, ret-x,1+x, ret)}^{(MS,MR)} \in V'_{\text{span class}2b}, 0 \leq ao \leq MR, \}
\]

\[
0 \leq mn \leq ret - x, 0 \leq an \leq 1 + x \}
\]

\[
= \{ M_{(2b,i),(0, ao, mn, an, ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn \leq ret - x, 0 \leq an \leq 1 + x, \}
\]

\[
0 \leq x \leq ret, 0 \leq ret \leq MR \}
\]

We can simplify the three inequalities that involve \( x \) in order to eliminate \( x \), although this is more complicated than in previous sublemmas. By summing the corresponding terms of the two inequalities, \( 0 \leq mn \leq ret - x \) and \( 0 \leq an \leq 1 + x \), gives \( 0 \leq mn + an \leq ret + 1 \). Intuitively, as \( x \) varies from 0 to \( ret \), the maximum possible value of \( an \) varies from 1 to \( ret + 1 \). However, the maximum possible value of \( mn \) varies from \( ret \) to 0. So although \( mn + an \leq ret + 1 \), we need an additional piece of information in the case where \( an = 0 \) to restrict \( mn \) to be less than or equal to \( ret \). Hence, the three inequalities involving \( x \) can be replaced by two inequalities, \( 0 \leq mn + an \leq ret + 1 \), and \( 0 \leq mn \leq ret \).

Performing this simplification means that \( V_{dc,\text{span class}2b} \) can be represented by the expression:

\[
V_{dc,\text{span class}2b} = \{ M_{(2b,i),(0, ao, mn, an, ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn \leq ret, \}
\]

\[
0 \leq mn + an \leq ret + 1, 0 \leq ret \leq MR \}
\]

(8.10)

from the definition of \( V_{(2b,i)}^{(MS,MR)} \) in row 3 of Table 8.1. Thus the sublemma is proved.

Figure 8.9 shows an abstract view of the spanning of all markings in \( V_{(1,i)}^{(MS,MR)} \), \( V_{(2a,i)}^{(MS,MR)} \), \( V_{(2b,i)}^{(MS,MR)} \), \( V_{(3a,i)}^{(MS,MR)} \), \( V_{(3b,i)}^{(MS,MR)} \), \( V_{(4,i)}^{(MS,MR)} \). For the situation where \( MR > 0 \), there are markings of class 4 that must be explored. The marking \( M_{(4,i),(MR,1,0,0,0,0)}^{(MS,MR)} \) can be reached from \( M_{(1,i),(MR,0,0,0,0)}^{(MS,MR)} \in V_{(1,i)}^{(MS,MR)} \) by occurrence of transition receive\_mess with binding \( <queue = [(i \otimes MS 1)^{MR-1}], sn = i \otimes MS 1, rn = i> \) (from Lemma 6.4 and the transition rule). This corresponds to the reception of an old duplicate message by the receiver. (When \( MR = 0 \), this particular transition will not be enabled by
8.4. A Spanning of all Markings

Figure 8.9: An abstract view of the spanning of all markings in $V_{(1,i)}$, $V_{(2a,i)}$, $V_{(3a,i)}$, and $V_{(3b,i)}$, with an arc from $M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$ to $M^{(MS,MR)}_{(4,i),(MR-1,0,0,0,0)}$ for the future spanning of all markings in $V_{(4,i)}$.

$V_{(3b,i)}^{(MS,MR)}$ because the message channel will be empty.) The next and final sublemma shows that all markings in $V_{(4,i)}^{(MS,MR)}$ are reachable from this same marking, $M^{(MS,MR)}_{(4,i),(MR-1,0,0,0,0)}$.

**Sublemma 8.1.6.** All markings in $V_{(4,i)}^{(MS,MR)}$ are reachable from $M^{(MS,MR)}_{(4,i),(MR-1,0,0,0,0)}$.

**Proof.** The proof of this sublemma is very similar to the proof of Sublemma 8.1.1. From Definition 7.6, $M^{(MS,MR)}_{(4,i),(MR-1,0,0,0,0)}$ represents the marking $M$ where:
8.4. A Spanning of all Markings

\[
\begin{align*}
M(\text{sender\_state}) &= 1'i \text{\_ready} & M(\text{send\_seq\_no}) &= 1'i \\
M(\text{receiver\_state}) &= 1'i \text{\_process} & M(\text{recv\_seq\_no}) &= 1'i \\
M(\text{mess\_channel}) &= 1'i[(i \cap MS 1)^{MR-1}] & M(\text{ack\_channel}) &= 1'i[] \\
M(\text{retrans\_counter}) &= 1'0
\end{align*}
\]

This marking enables the send\_ack transition with binding \(<\text{queue} = [], rn = i>\) (Lemma 6.5). Occurrences of send\_ack and receive\_mess (Lemmas 6.5 and 6.4 and the transition rule) can be repeated \(MR - 1\) number of times, i.e., until the message channel is empty. Hence, the marking \(M^{(MS,MR)}_{(4,i),(MR-1,0,0,0,0)}\) can reach the markings in \(V_{\text{span\_class\_4}}\):

\[
V_{\text{span\_class\_4}} = \{M^{(MS,MR)}_{(4,i),(MR-1-x,x,0,0,0,0)} \mid 0 \leq x \leq MR - 1, MR > 0\}
\] (8.11)

Again, the downward-closed set of markings of all markings in \(V_{\text{span\_class\_4}}\) can be reached by repeated successive firing of mess\_loss and ack\_loss (by Lemmas 6.8 and 6.9, respectively). From Equation (8.11) and Definition 7.11 this set of markings is:

\[
V_{dc,\text{span\_class\_4}} = DC(V_{\text{span\_class\_4}})
\]

\[
= \{M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0,0)} \mid M^{(MS,MR)}_{(4,i),(MR-1-x,x,0,0,0,0)} \in V_{\text{span\_class\_4}}, 0 \leq mo \leq MR - 1 - x, \\
0 \leq ao \leq x \}
\]

\[
= \{M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0,0)} \mid 0 \leq mo \leq MR - 1 - x, 0 \leq ao \leq x, 0 \leq x \leq MR - 1, \\
MR > 0\}
\]

The inequalities involving \(mo, ao\) and \(x\) can be simplified in a similar way to Sublemma 8.1.1. Summing the corresponding terms from 0 \(\leq mo \leq MR - 1 - x\) and 0 \(\leq ao \leq x\), we get 0 \(\leq mo + ao \leq MR - 1\). Intuitively, as \(x\) varies from 0 to \(MR - 1\), the maximum possible value of \(mo\) varies from \(MR - 1\) to 0, and the maximum possible value of \(ao\) varies from 0 to \(MR - 1\). Hence, the sum of \(mo\) and \(ao\) can never be greater than \(MR - 1\). The inequality 0 \(\leq x \leq MR - 1\) becomes redundant. Performing this simplification results in the expression:

\[
V_{dc,\text{span\_class\_4}} = \{M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0,0)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq ao \leq MR - 1, MR > 0\}
\] (8.12)

from the definition of \(V^{(MS,MR)}_{(4,i)}\) in row 6 of Table 8.1. Thus the sublemma is proved. □

Sublemmas 8.1.1 to 8.1.6 show that for any 0 \(\leq i \leq MS\), all markings in \(V^{(MS,MR)}_{i}\) are reachable from \(M^{(MS,MR)}_{(1,i),(MR,0,0,0,0,0)}\). This is shown in Fig. 8.10 for one specific value of \(i\). The concrete case where \(i = MS\) for any \(MS \in \mathbb{N}^+\) is no different, as the modulo addition operator simply wraps sequence numbers back to 0.
8.4. A Spanning of all Markings

The final step of the proof of this lemma is to note that from Lemma 6.6, \( M_{(2b,i)}^{(MS,MR)} \in V_{(2b,i)}^{(MS,MR)} \) enables the receive_ack transition with the binding \(<queue = [], sn = i, rn = i \oplus MS 1, rc = MR>\). From the transition rule, this results in the marking \( M_{(1,i)}^{(MS,MR)} \in V_{(1,i\oplus MS 1)}^{(MS,MR)} \). This occurrence of receive_ack corresponds to reception of the first acknowledgement for the new message (the message with sequence number \( i \)). This is the acknowledgement for which the sender is waiting. When this transition occurs, the retransmission counter is reset to 0, the sender sequence number

Figure 8.10: An abstract view of the complete spanning of all markings in \( V_i^{(MS,MR)} \), i.e. all markings in \( \bigcup_{class \in \{1,2a,2b,3a,3b,4\}} V_{(class,i)}^{(MS,MR)} \).
is incremented to \( i \oplus_{MS} 1 \), and the messages that were considered ‘new’ in \( M^{(MS,MR)}_{(2b,1),(0,0,MR+1,MR)} \) now become old duplicates. This is shown in Fig. 8.11, where the spanning of \( V_i^{(MS,MR)} \) is shown on the left with a solid outline and the spanning of \( V_{i+\oplus_{MS} 1}^{(MS,MR)} \) is shown on the right with a dashed outline. The new arc, in bold, connects the spanning of \( V_i^{(MS,MR)} \) with that of \( V_{i+\oplus_{MS} 1}^{(MS,MR)} \). We conclude that for all \( MS \in \mathbb{N}^+ \), and \( 0 \leq i \leq MS \), all markings in \( V_i^{(MS,MR)} \cup \{ M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)} \} \) are reachable from \( M^{(MS,MR)}_{(1,1),(MR,0,0,0,0)} \) and the Spanning Lemma, Lemma 8.1, is proved.

We now show that all markings in \( \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \) are reachable from \( M^{(MS,MR)}_{(1,0),(MR,0,0,0,0)} \).

**Lemma 8.2.** \( \forall M \in \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)}, M \in [M^{(MS,MR)}_{(1,0),(MR,0,0,0,0)}, M^{(MS,MR)}_{(1,1),(MR,0,0,0,0)}] \).

**Proof.** The spanning lemma shows that for any \( i \in \{0, 1, ..., MS\} \) we have that

\[
\forall M \in V_i^{(MS,MR)}, M \in [M^{(MS,MR)}_{(1,0),(MR,0,0,0,0)}] \text{ and } M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)} \in [M^{(MS,MR)}_{(1,1),(MR,0,0,0,0)}]
\]

Hence, we have that

\[
\forall M \in V_0^{(MS,MR)}, M \in [M^{(MS,MR)}_{(1,0),(MR,0,0,0,0)}] \text{ and } \forall M \in V_1^{(MS,MR)}, M \in [M^{(MS,MR)}_{(1,1),(MR,0,0,0,0)}]
\]

Because \( M^{(MS,MR)}_{(1,1),(MR,0,0,0,0)} \in [M^{(MS,MR)}_{(1,0),(MR,0,0,0,0)}] \) (from Lemma 8.1), then

\[
\forall M \in V_0^{(MS,MR)} \cup V_1^{(MS,MR)}, M \in [M^{(MS,MR)}_{(1,0),(MR,0,0,0,0)}]
\]

From successive applications of the spanning lemma we then obtain that all markings in \( \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \) are reachable from \( M^{(MS,MR)}_{(1,0),(MR,0,0,0,0)} \). (This is illustrated by Fig. 8.12.) Hence, the lemma is proved.

**Corollary 8.1.** \( \forall i, j \in \{0, 1, ..., MS\}, M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)} \in [M^{(MS,MR)}_{(1,j),(MR,0,0,0,0)}] \).

Thus all markings in \( \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \) are also reachable from \( M^{(MS,MR)}_{(1,1),(MR,0,0,0,0)} \) because \( M^{(MS,MR)}_{(1,0),(MR,0,0,0,0)} \in [M^{(MS,MR)}_{(1,1),(MR,0,0,0,0)}] \) from the above corollary. To finalise the proof we prove the following lemma.

**Lemma 8.3.** The marking \( M^{(MS,MR)}_{(1,1),(MR,0,0,0,0)} \) is reachable from the initial marking, \( M^{(MS,MR)}_{(1,0),(0,0,0,0,0)} \).

**Proof.** Proof is by direct inspection of the CPN diagram in Fig. 5.5. The initial marking enables transition \texttt{send\_mess} with binding \( <queue = [], sn = 0> \). This results in the marking \( M^{(MS,MR)}_{(2a,0),(0,0,1,0,0)} \). From this marking, transition \texttt{timeout\_retrans} can occur consecutively \( MR \) number of times, i.e. retransmitting the message with sequence number 0 until the maximum number of retransmissions is reached. The resultant marking is \( M^{(MS,MR)}_{(2a,0),(0,0,MR+1,0,MR)} \) in which \( MR + 1 \) copies of the message with sequence number 0 are in the message channel. From this marking, \texttt{receive\_mess} can occur with binding
8.4. A Spanning of all Markings

Figure 8.11: An abstract view of the complete spanning of all markings in $V_{i}^{(MS,MR)}$, i.e. all markings in $\bigcup_{class \in \{1,2a,2b,3a,3b,4\}} V_{class,i}^{(MS,MR)}$. 
8.5. Containment of all Successors

(queue = [0]^MR), sn = 0, rn = 0) resulting in marking M_{(3b,0),(0,0,MR,0,MR)}^{(MS,MR)}. From this marking, send_ack can occur with the binding <queue = [], rn = 1> leading to M_{(2b,0),(0,0,MR,1,MR)}^{(MS,MR)}.

The single acknowledgement with sequence number 1 is the acknowledgement for which the sender is waiting. The occurrence of receive_ack with binding <queue = [], sn = 0, rn = 1, rc = MR> from M_{(2b,0),(0,0,MR,1,MR)} leads to the marking M_{(1,1),(MR,0,0,0,0)}^{(MS,MR)} in which the message with sequence number 0 has been acknowledged, the sender sequence number has been incremented to 1, the retransmission counter of the sender has been reset to 0, and the duplicate messages with sequence number 0 are now considered as old messages. Thus M_{(1,0),(0,0,0,0,0)}^{(MS,MR)} can reach M_{(1,1),(MR,0,0,0,0)}^{(MS,MR)} and the lemma is proved.

Lemma 8.3 shows that the initial marking can reach M_{(1,0),(0,0,0,0,0)}^{(MS,MR)} which can in turn, from Lemma 8.2 and the spanning lemma, reach all markings in \( \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \). Hence, Part A of the proof of Theorem 8.1 is complete.

8.5 Containment of all Successors

The second part of the proof of Theorem 8.1 is provided by the successor lemma. Before we state and prove the successor lemma, we state and prove a series of propositions about simplifying inequalities.

**Proposition 8.1.** For 0 \leq i \leq MS and class \in \{1, 2a, 2b, 3a, 3b, 4\},

\[
V_a = \{M_{(class,i),(mo-1,ao,mm,an,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, mo \geq 1, MR \geq 1\}
\]

can be expressed more simply as

\[
V_b = \{M_{(class,i),(mo,ao,mm,an,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, MR \geq 1\},
\]

where \( V_a = V_b \).

**Proof.** Because \( mo \geq 1 \), then 1 \leq mo \leq MR and hence ao is restricted to 0 \leq ao \leq MR - 1. Subtracting 1 from each term in 1 \leq mo \leq MR and \( mo \geq 1 \) gives 0 \leq mo - 1 \leq MR - 1 and \( mo - 1 \geq 0 \). Hence the inequalities of \( V_a \) can be rewritten as 0 \leq mo - 1 + ao \leq MR - 1 and \( mo - 1 \geq 0 \), and \( V_a \) can be written as \( \{M_{(class,i),(mo-1,ao,mm,an,ret)}^{(MS,MR)} \mid 0 \leq mo - 1 + ao \leq MR - 1, mo - 1 \geq 0, MR \geq 1\} \). Substituting \( mo' = mo - 1 \) gives \( \{M_{(class,i),(mo',ao,mm,an,ret)}^{(MS,MR)} \mid 0 \leq mo' + ao \leq MR - 1, mo' \geq 0, MR \geq 1\} \). The inequality, \( mo' \geq 0 \), is now redundant. Deleting this and removing the prime from \( mo' \) gives \( \{M_{(class,i),(mo,ao,mm,an,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, MR \geq 1\} \), which equals \( V_b \), hence the proposition is proved.

**Proposition 8.2.** For 0 \leq i \leq MS and class \in \{1, 2a, 2b, 3a, 3b, 4\},

\[
V_a = \{M_{(class,i),(mo,ao,mm,an,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, ao \geq 1, MR \geq 1\}
\]

can be expressed more simply as

\[
V_b = \{M_{(class,i),(mo,ao,mm,an,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, MR \geq 1\},
\]

where \( V_a = V_b \).

**Proof.** This proof is identical to the proof of Proposition 8.1 except that the roles of \( mo \) and \( ao \) are reversed.
Figure 8.12: An abstract view of the complete spanning of the markings in $\bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)}$. 
8.5. Containment of all Successors

**Proposition 8.3.** For $0 \leq i \leq MS$ and class $\in \{1, 2a, 2b, 3a, 3b, 4\}$,

$$V_a = \{M_{(class,i),(mo,ao,an-1,an,ret)}^{(MS,MR)} \mid 1 \leq mn \leq ret + 1, 0 \leq ret \leq MR\}$$

can be expressed more simply as

$$V_b = \{M_{(class,i),(mo,ao,an,ret)}^{(MS,MR)} \mid 0 \leq mn \leq ret, 0 \leq ret \leq MR\}, \text{ where } V_a = V_b.$$

**Proof.** Subtracting 1 from each term in $1 \leq mn \leq ret + 1$ gives $0 \leq mn - 1 \leq ret$, and thus $V_a$ can be written as

$$\{M_{(class,i),(mo,ao,an-1,an,ret)}^{(MS,MR)} \mid 0 \leq mn - 1 \leq ret, 0 \leq ret \leq MR\}.$$ Substituting $mn' = mn - 1$ into this equation gives

$$\{M_{(class,i),(mo,ao,an',an,ret)}^{(MS,MR)} \mid 0 \leq mn' \leq ret, 0 \leq ret \leq MR\}.$$ Removing the prime from $mn'$ gives

$$\{M_{(class,i),(mo,ao,an,ret)}^{(MS,MR)} \mid 0 \leq mn \leq ret, 0 \leq ret \leq MR\},$$

which equals $V_b$, hence the proposition is proved. \hfill \square

**Proposition 8.4.** For $0 \leq i \leq MS$ and class $\in \{1, 2a, 2b, 3a, 3b, 4\}$,

$$V_a = \{M_{(class,i),(mo,ao,an-1,an,ret)}^{(MS,MR)} \mid 1 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1, 0 < ret \leq MR\}$$

can be expressed more simply as

$$V_b = \{M_{(class,i),(mo,ao,an,ret)}^{(MS,MR)} \mid 0 \leq mn \leq ret - 1, 0 \leq mn + an \leq ret, 0 < ret \leq MR\}, \text{ where } V_a = V_b.$$

**Proof.** Because $1 \leq mn \leq ret$, then $an$ is restricted to $0 \leq an \leq ret$. Subtracting 1 from each term in $1 \leq mn \leq ret$ gives $0 \leq mn - 1 \leq ret - 1$. Hence the inequalities of $V_a$ can be rewritten as $0 \leq mn - 1 \leq ret - 1$ and $0 \leq mn + an - 1 \leq ret$, and $V_a$ can be written as

$$\{M_{(class,i),(mo,ao,an-1,an,ret)}^{(MS,MR)} \mid 0 \leq mn - 1 \leq ret - 1, 0 \leq mn + an - 1 \leq ret, 0 < ret \leq MR\}.$$ Substituting $mn' = mn - 1$ into this equation gives

$$\{M_{(class,i),(mo,ao,an',an,ret)}^{(MS,MR)} \mid 0 \leq mn' \leq ret - 1, 0 \leq mn' + an \leq ret, 0 < ret \leq MR\}.$$ Removing the prime from $mn'$ gives

$$\{M_{(class,i),(mo,ao,an,ret)}^{(MS,MR)} \mid 0 \leq mn \leq ret - 1, 0 \leq mn + an \leq ret, 0 < ret \leq MR\},$$

which equals $V_b$, hence the proposition is proved. \hfill \square

**Proposition 8.5.** For $0 \leq i \leq MS$ and class $\in \{1, 2a, 2b, 3a, 3b, 4\}$,

$$V_a = \{M_{(class,i),(mo,ao,an-1,an,ret)}^{(MS,MR)} \mid 0 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1, an \geq 1\}$$

can be expressed more simply as

$$V_b = \{M_{(class,i),(mo,ao,an,ret)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret\}, \text{ where } V_a = V_b.$$

**Proof.** Because $an \geq 1$, then $1 \leq an \leq ret + 1$, but this does not restrict the value of $mn$, which is $0 \leq mn \leq ret$. Hence, $1 \leq mn + an \leq ret + 1$. Subtracting 1 from each term in $1 \leq mn + an \leq ret + 1$ and $an \geq 1$ gives $0 \leq mn + an - 1 \leq ret$ and $an - 1 \geq 0$, hence $V_a$ can be rewritten as

$$\{M_{(class,i),(mo,ao,an-1,an,ret)}^{(MS,MR)} \mid 0 \leq mn \leq ret, 0 \leq mn + an - 1 \leq ret, an - 1 \geq 0\}.$$ Substituting $an' = an - 1$ gives

$$\{M_{(class,i),(mo,ao,an',ret)}^{(MS,MR)} \mid 0 \leq mn \leq ret, 0 \leq mn + an' \leq ret, an' \geq 0\}.$$ The inequalities $0 \leq mn \leq ret$ and $an' \geq 0$ have become redundant. Deleting them and dropping the prime on $an'$ gives

$$\{M_{(class,i),(mo,ao,an,ret)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret\},$$

which equals $V_b$, hence the proposition is proved. \hfill \square

**Proposition 8.6.** For $0 \leq i \leq MS$ and class $\in \{1, 2a, 2b, 3a, 3b, 4\}$,
8.5. Containment of all Successors

\[ V_a = \{ M_{(class,i),(mo,ao,mn,an-1,ret)}^{(MS,MR)} | 0 \leq mn + an \leq \text{ret}, \text{ret} \geq 1, an \geq 1 \} \]

can be expressed more simply as

\[ V_b = \{ M_{(class,i),(mo,ao,mn,an,ret)}^{(MS,MR)} | 0 \leq mn + an \leq \text{ret} - 1, \text{ret} \geq 1 \}, \]

where \( V_a = V_b \).

**Proof.** Because \( an \geq 1 \), then \( 1 \leq an \leq \text{ret} \), which restricts the value of \( mn \) to \( 0 \leq mn \leq \text{ret} - 1 \).

Subtracting 1 from each term in \( an \geq 1 \) and \( 1 \leq an \leq \text{ret} \) gives \( an - 1 \geq 0 \) and \( 0 \leq an - 1 \leq \text{ret} - 1 \). Hence, the inequalities can be rewritten as \( 0 \leq mn + an - 1 \leq \text{ret} - 1 \) and \( an - 1 \geq 0 \), giving

\[ \{ M_{(class,i),(mo,ao,mn,an-1,ret)}^{(MS,MR)} | 0 \leq mn + an - 1 \leq \text{ret} - 1, an - 1 \geq 0 \}. \]

Substituting \( an' = an - 1 \) gives \( \{ M_{(class,i),(mo,ao,mn,an',ret)}^{(MS,MR)} | 0 \leq mn + an' \leq \text{ret} - 1, an' \geq 0 \}. \) The inequality \( an' \geq 0 \) becomes redundant. Deleting it and dropping the prime on \( an' \) gives \( \{ M_{(class,i),(mo,ao,mn,an,ret)}^{(MS,MR)} | 0 \leq mn + an \leq \text{ret} - 1 \}, \) which equals \( V_b \), hence the proposition is proved.

**Proposition 8.7.** For \( 0 \leq i \leq MS \) and \( class \in \{1, 2a, 2b, 3a, 3b, 4\} \),

\[ V_a = \{ M_{(class,i),(mo,ao,mn+1,an,ret+1)}^{(MS,MR)} | 0 \leq mn \leq \text{ret} + 1, 0 \leq \text{ret} < MR \} \]

can be expressed more simply as

\[ V_b = \{ M_{(class,i),(mo,ao,mn,an,ret)}^{(MS,MR)} | 1 \leq mn \leq \text{ret} + 1, 1 \leq \text{ret} \leq MR \}, \]

where \( V_a = V_b \).

**Proof.** Adding 1 to each term in \( 0 \leq mn \leq \text{ret} + 1 \) and \( 0 \leq \text{ret} < MR \) from \( V_a \) gives \( 1 \leq mn + 1 \leq \text{ret} + 2 \) and \( 1 \leq \text{ret} + 1 \leq MR \), respectively. Substituting these inequalities into \( V_a \) gives

\[ \{ M_{(class,i),(mo,ao,mn+1,an,ret+1)}^{(MS,MR)} | 1 \leq mn + 1 \leq \text{ret} + 2, 1 \leq \text{ret} + 1 \leq MR \}. \]

Substituting \( mn' = mn + 1 \) and \( \text{ret}' = \text{ret} + 1 \) into this equation gives \( \{ M_{(class,i),(mo,ao,mn',an,ret')}^{(MS,MR)} | 1 \leq mn' \leq \text{ret}' + 1, 1 \leq \text{ret}' \leq MR \}. \) Dropping the prime from \( mn' \) and \( \text{ret}' \) gives \( \{ M_{(class,i),(mo,ao,mn,an,ret)}^{(MS,MR)} | 1 \leq mn \leq \text{ret} + 1, 1 \leq \text{ret} \leq MR \}, \) which equals \( V_b \), hence the proposition is proved.

**Proposition 8.8.** For \( 0 \leq i \leq MS \) and \( class \in \{1, 2a, 2b, 3a, 3b, 4\} \),

\[ V_a = \{ M_{(class,i),(mo,ao,mn+1,an,ret+1)}^{(MS,MR)} | 0 \leq mn \leq \text{ret}, 0 \leq mn + an \leq \text{ret} + 1, 0 \leq \text{ret} < MR \}

can be expressed more simply as

\[ V_b = \{ M_{(class,i),(mo,ao,mn,an,ret)}^{(MS,MR)} | 1 \leq mn \leq \text{ret}, 1 \leq mn + an \leq \text{ret} + 1, 1 \leq \text{ret} \leq MR \}, \]

where \( V_a = V_b \).

**Proof.** Adding 1 to each term in \( 0 \leq mn \leq \text{ret}, 0 \leq mn + an \leq \text{ret} + 1 \) and \( 0 \leq \text{ret} < MR \) from \( V_a \) gives \( 1 \leq mn + 1 \leq \text{ret} + 1 \) and \( 1 \leq \text{ret} + 1 \leq MR \), respectively. Substituting these inequalities into \( V_a \) gives

\[ \{ M_{(class,i),(mo,ao,mn,an+1,ret+1)}^{(MS,MR)} | 1 \leq mn + 1 \leq \text{ret} + 1, 1 \leq \text{ret} + 1 \leq MR \}. \]

Substituting \( mn' = mn + 1 \) and \( \text{ret}' = \text{ret} + 1 \) into this equation gives \( \{ M_{(class,i),(mo,ao,mn',an,ret')}^{(MS,MR)} | 1 \leq mn' \leq \text{ret}' + 1, 1 \leq \text{ret}' \leq MR \}. \) Dropping the prime from \( mn' \) and \( \text{ret}' \) gives \( \{ M_{(class,i),(mo,ao,mn,an,ret)}^{(MS,MR)} | 1 \leq mn \leq \text{ret} + 1, 1 \leq \text{ret} \leq MR \}, \) which equals \( V_b \), hence the proposition is proved.

We now state and prove the successor lemma.
8.5. Containment of all Successors

Lemma 8.4. Successor Lemma. For every \( MR \in \mathbb{N}, MS \in \mathbb{N}^+ \), and for \( 0 \leq i \leq MS \), every enabled binding element of every marking in \( \mathcal{V}_i^{(MS,MR)} \) is described exactly by \( A_i^{(MS,MR)} \) and every successor of every marking in \( \mathcal{V}_i^{(MS,MR)} \) is contained in \( \bigcup_{0 \leq i \leq MS} \mathcal{V}_i^{(MS,MR)} \).

Proof. This proof, like the proof of the spanning lemma, is divided into 6 sublemmas, one for each of the 6 subsets of markings in \( \mathcal{V}_i^{(MS,MR)} \) corresponding to the 6 classes of marking. Recall that it has been proved that all markings in \( \bigcup_{0 \leq i \leq MS} \mathcal{V}_i^{(MS,MR)} \) are reachable from the initial marking, hence by Lemma 6.1, \( \forall p \in P, \forall M \in \bigcup_{0 \leq i \leq MS} \mathcal{V}_i^{(MS,MR)}, |M(p)| = 1 \). Lemmas 6.2 to 6.9, capturing the necessary characteristics for a marking to enable a transition, and Propositions 8.1 to 8.8, for simplifying inequalities, form the basis for the proofs of the following 6 sublemmas.

Sublemma 8.4.1. All enabled binding elements of all markings in \( \mathcal{V}_i^{(MS,MR)} \) are described exactly by \( A_i^{(MS,MR)} \) in Table 8.2 and all immediate successors of all markings in \( \mathcal{V}_i^{(MS,MR)} \) are contained in \( \mathcal{V}_i^{(MS,MR)} \cup \mathcal{V}_{(2a,i)}^{(MS,MR)} \cup \mathcal{V}_{(4,i)}^{(MS,MR)} \).

Proof. From Definition 7.6 and row 1 of Table 8.1, the markings in \( \mathcal{V}_i^{(MS,MR)} \) have the form \( M_{i}^{(MS,MR)}(mo,ao,0,0,0) \) where, for a marking \( M \in \mathcal{V}_i^{(MS,MR)} \):

\[
\begin{align*}
M(\text{sender\_state}) &= 1^i \text{send ready} & M(\text{send\_seq\_no}) &= 1^i \\
M(\text{receiver\_state}) &= 1^i \text{recv ready} & M(\text{recv\_seq\_no}) &= 1^i \\
M(\text{mess\_channel}) &= 1^i ([i \ominus MS 1]^{mo}) & M(\text{ack\_channel}) &= 1^i [i^{ao}] \\
M(\text{retrans\_counter}) &= 1^i 0
\end{align*}
\]

where \( 0 \leq mo + ao \leq MR \). We now systematically consider the enabling and occurrence of each transition from the markings in \( \mathcal{V}_i^{(MS,MR)} \).

**send**: From Lemma 6.2, all markings in \( \mathcal{V}_i^{(MS,MR)} \) enable send, with binding <queue = \( ([i \ominus MS 1]^{mo}), sn = i > \). The resulting set of outgoing arcs is given by:

\[
\{ (M_{i}^{(MS,MR)}(1,i),(mo,ao,0,0,0), \text{send\_mess} < \text{queue} = ([i \ominus MS 1]^{mo}), sn = i >, M_{(2a,i)}^{(MS,MR)}(mo,ao,1,0,0) ) | M_{(1,i)}^{(MS,MR)}(mo,ao,0,0,0) \in \mathcal{V}_i^{(MS,MR)} \}
\]

This set of arcs corresponds exactly to the arcs defined by row 1 of Table 8.2. After substituting the restrictions on \( mo \) and \( ao \) from the definition of \( \mathcal{V}_i^{(MS,MR)} \), the set of destination markings is given by \( \{ M_{(2a,i)}^{(MS,MR)}(mo,ao,1,0,0) | 0 \leq mo + ao \leq MR \} \). By inspection of row 2 of Table 8.1, this is a subset of \( \mathcal{V}_{(2a,i)}^{(MS,MR)} \).

**mess**: From Lemma 6.8, all markings in \( \{ M_{(1,i)}^{(MS,MR)}(1,i),(mo,ao,0,0,0) \in \mathcal{V}_i^{(MS,MR)} | M_{(1,i)}^{(MS,MR)}(mo,ao,0,0,0) \in \mathcal{V}_i^{(MS,MR)} \}, mo \geq 1 \} \) (where \( mo \geq 1 \) implies \( MR \geq 1 \) because \( ao \) cannot be negative) enable mess.
with binding 

\[
\text{\textless queue} = \left[ (i \oplus_{MS} 1)^{mo} \right], \text{sn} = i \oplus_{MS} 1 >
\]

The resulting set of outgoing arcs is given by:

\[
\{(M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} \cdot \text{mess} < \text{queue} = \left[ (i \oplus_{MS} 1)^{mo} \right], \text{sn} = i \oplus_{MS} 1 >, \text{m} \geq 1, M \geq 1 \}
\]

which corresponds exactly to the set of arcs defined by row 2 of Table 8.2. After substituting the restrictions on \( mo \) and \( ao \) from the definition of \( V_{(1,i)}^{(MS,MR)} \), the destination markings are given by

\[
\{(M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} | 0 \leq mo + ao \leq MR, mo \geq 1, MR \geq 1 \}
\]

From Proposition 8.1, this set of destination markings can be expressed more simply as

\[
\{(M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} | 0 \leq mo + ao \leq MR - 1, MR \geq 1 \}
\]

which, by inspection of row 1 of Table 8.1, is a subset of \( V_{(1,i)}^{(MS,MR)} \).

**receive**,** mess:** From Lemma 6.4, all markings in \( \{M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} | M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} \notin V_{(1,i)}^{(MS,MR)}, mo \geq 1 \} \) (where \( mo \geq 1 \) implies \( MR \geq 1 \) because \( ao \) cannot be negative) also enable receive,** mess,** with binding 

\[
\text{\textless queue} = \left[ (i \oplus_{MS} 1)^{mo-1} \right], \text{sn} = i \oplus_{MS} 1, rn = i >.
\]

The resulting set of outgoing arcs is given by:

\[
\{(M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} \cdot \text{receive} < \text{queue} = \left[ (i \oplus_{MS} 1)^{mo-1} \right], \text{sn} = i \oplus_{MS} 1, rn = i >, \text{m} \geq 1, M \geq 1 \}
\]

which corresponds exactly to the set of arcs defined by row 3 of Table 8.2. After substituting the restrictions on \( mo \) and \( ao \) from the definition of \( V_{(1,i)}^{(MS,MR)} \), the destination markings are given by

\[
\{(M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} | 0 \leq mo + ao \leq MR, mo \geq 1, MR \geq 1 \}
\]

From Proposition 8.1, we have

\[
\{(M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} | 0 \leq mo + ao \leq MR - 1, MR \geq 1 \}
\]

which equals \( V_{(1,i)}^{(MS,MR)} \) from row 6 of Table 8.1.

**ack**,** loss:** From Lemma 6.9, all markings in \( \{M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} | M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} \notin V_{(1,i)}^{(MS,MR)}, ao \geq 1 \} \) (where \( ao \geq 1 \) implies \( MR \geq 1 \) because \( mo \) cannot be negative) enable \( \text{ack} < \text{loss}, \) with binding 

\[
\text{\textless queue} = \left[ i^{ao} \right], rn = i >.
\]

The resulting set of outgoing arcs is given by:

\[
\{(M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} \cdot \text{ack} < \text{loss} = \left[ i^{ao} \right], RN = i >, M_{(1,i),(mo,ao,0,0,0)}^{(MS,MR)} \notin V_{(1,i)}^{(MS,MR)}, ao \geq 1, M \geq 1 \}
\]
which corresponds exactly to the set of arcs defined by row 4 of Table 8.2. After substituting the restrictions on \( mo \) and \( ao \) from the definition of \( V^{(MS,MR)}_{(1,i)} \), the destination markings are given by

\[
\{ M^{(MS,MR)}_{(1,i),(mo,ao-1,0,0,0)} \mid 0 \leq mo + ao \leq MR, ao \geq 1, MR \geq 1 \}
\]

From Proposition 8.2, we have

\[
\{ M^{(MS,MR)}_{(1,i),(mo,ao,0,0,0)} \mid 0 \leq mo + ao \leq MR - 1, MR \geq 1 \}
\]

which, by inspection of row 1 of Table 8.1, is a subset of \( V^{(MS,MR)}_{(1,i)} \).

**receive_dup_ack:** From Lemma 6.7, all markings in \( \{ M^{(MS,MR)}_{(1,i),(mo,ao,0,0,0)} \mid M^{(MS,MR)}_{(1,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(1,i)}, ao \geq 1 \} \) (where \( ao \geq 1 \) implies \( MR \geq 1 \) because \( mo \) cannot be negative) also enable **receive_dup_ack**, with binding \( <queue = [i^{ao-1}], sn = i, rn = i> \). The resulting set of outgoing arcs is given by:

\[
\{ (M^{(MS,MR)}_{(1,i),(mo,ao,0,0,0)}, receive\_dup\_ack <queue = [i^{ao-1}], sn = i, rn = i>, M^{(MS,MR)}_{(1,i),(mo,ao-1,0,0,0)} \mid M^{(MS,MR)}_{(1,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(1,i)}, ao \geq 1, MR \geq 1 \}
\]

which corresponds exactly to the set of arcs defined by row 5 of Table 8.2. After substituting the restrictions on \( mo \) and \( ao \) from the definition of \( V^{(MS,MR)}_{(1,i)} \), the destination markings are given by

\[
\{ M^{(MS,MR)}_{(1,i),(mo,ao-1,0,0,0)} \mid 0 \leq mo + ao \leq MR, ao \geq 1, MR \geq 1 \}
\]

which is identical to the set of destination markings of the **ack_loss** transition above. These have already been shown to be a subset of \( V^{(MS,MR)}_{(1,i)} \).

**timeout_retrans, send_ack and receive_ack:** From Lemmas 6.3, 6.5 and 6.6, none of these three transitions is enabled by any marking in \( V^{(MS,MR)}_{(1,i)} \). This corresponds to the absence of these three transitions from Table 8.2.

The binding elements enabled by the markings in \( V^{(MS,MR)}_{(1,i)} \) correspond exactly to those presented in Table 8.2. The successor marking of each enabled binding element is a member of \( V^{(MS,MR)}_{(1,i)} \cup V^{(MS,MR)}_{(2a,i)} \cup V^{(MS,MR)}_{(4,i)} \). Thus the sublemma is proved.

**Sublemma 8.4.2.** All enabled binding elements of all markings in \( V^{(MS,MR)}_{(2a,i)} \) are described exactly by \( M^{(MS,MR)}_{(2a,i),(mo,ao,mn,0,ret)} \) in Table 8.3 and all immediate successors of all markings in \( V^{(MS,MR)}_{(2a,i)} \) are contained in

\[
V^{(MS,MR)}_{(2a,i)} \cup V^{(MS,MR)}_{(3a,i)} \cup V^{(MS,MR)}_{(3b,i)}
\]

**Proof:** From Definition 7.6 and row 2 of Table 8.1, the markings in \( V^{(MS,MR)}_{(2a,i)} \) have the form \( M^{(MS,MR)}_{(2a,i),(mo,ao,mn,0,ret)} \) where, for a marking \( M \in V^{(MS,MR)}_{(2a,i)} \):
8.5. Containment of all Successors

\[
M(\text{sender\_state}) = 1^\\text{wait} \cdot \text{ack} \quad M(\text{send\_seq\_no}) = 1^i \\
M(\text{receiver\_state}) = 1^r \cdot \text{ready} \quad M(\text{recv\_seq\_no}) = 1^i \\
M(\text{mess\_channel}) = 1^i[(i \odot MS 1)^{mo \cdot i^{mn}}] \quad M(\text{ack\_channel}) = 1^i[i^{ao}] \\
M(\text{retrans\_counter}) = 1^r \text{ret}
\]

where \(0 \leq mo + ao \leq MR, 0 \leq mn \leq ret + 1\) and \(0 \leq ret \leq MR\). Considering each transition in turn for markings in \(V_{(2a,i)}^{(MS,MR)}\):

**timeout\_retrans:** From Lemma 6.3, all markings in \(\{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \in V_{(2a,i)}^{(MS,MR)}, ret < MR\}\) enable **timeout\_retrans**, with binding \(<\text{queue} = [(i \odot MS 1)^{mo \cdot i^{mn}}], sn = i, rc = ret\>_i\). The resulting set of outgoing arcs is given by:

\[
\{M_{(2a,i),(mo,ao,mn+1,0,ret+1)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq ret + 1, 0 \leq ret < MR\}
\]

From Proposition 8.7, we have

\[
\{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq ret + 1, 0 \leq ret < MR\}
\]

which, by inspection of row 2 of Table 8.1, is a subset of \(V_{(2a,i)}^{(MS,MR)}\).

**mess\_loss:** From Lemma 6.8, all markings in \(\{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \in V_{(2a,i)}^{(MS,MR)}, mo + mn \geq 1\}\) satisfy the enabling conditions of **mess\_loss**, with binding \(<\text{queue} = [(i \odot MS 1)^{mo \cdot i^{mn}}], sn = i \odot MS 1\>_i\) provided \(mo \geq 1\) (implying that \(MR \geq 1\)) or \(<\text{queue} = [(i \odot MS 1)^{mo \cdot i^{mn}}], sn = i\>_i\) provided \(mn \geq 1\). The resulting sets of outgoing arcs are given by:

\[
\{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 1 \leq mn \leq ret + 1, 1 \leq ret \leq MR\}
\]

and

\[
\{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR, 1 \leq mn \leq ret + 1, 1 \leq ret \leq MR\}
\]
which correspond exactly to the sets of arcs defined by rows 2 and 3 respectively of Table 8.3.

After substituting the restrictions on \(mo, ao, mn\) and \(ret\) from the definition of \(V_{(2a,i)}^{(MS, MR)}\) the destination markings are given by

\[
\{M_{(2a,i),(mo-1,ao,mn,0,ret)}^{(MS, MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR, mo \geq 1, MR \geq 1 \}
\]

and

\[
\{M_{(2a,i),(mo,ao,mn-1,0,ret)}^{(MS, MR)} \mid 0 \leq mo + ao \leq MR, 1 \leq mn \leq ret + 1, 0 \leq ret \leq MR \}
\]

respectively. From Proposition 8.1, the first set becomes

\[
\{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS, MR)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR, MR \geq 1 \}
\]

and from Proposition 8.3 the second set becomes

\[
\{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS, MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq ret, 0 \leq ret \leq MR \}
\]

By inspection of Table 8.1, both sets of destination markings are subsets of \(V_{(2a,i)}^{(MS, MR)}\).

**receive**\_mess: From Lemma 6.4, all markings in \(\{M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS, MR)} \mid M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS, MR)} \in V_{(2a,i)}^{(MS, MR)}, mo \geq 1 \}\) and in \(\{M_{(2a,i),(0,ao,mn,0,ret)}^{(MS, MR)} \mid M_{(2a,i),(0,ao,mn,0,ret)}^{(MS, MR)} \in V_{(2a,i)}^{(MS, MR)}, mn \geq 1 \}\) enable receive mess, with binding \(<\text{queue} = [(i \ominus MS)_{1}^{mo-1}, i^{mn}], sn = i \ominus MS 1, rn = i>\) and \(<\text{queue} = [i^{mn-1}], sn = i, rn = i>\) respectively. The resulting sets of outgoing arcs are given by:

\[
\{(M_{(2a,i),(mo,ao,mn,0,ret)}^{(MS, MR)}), \text{receive} \_ \text{mess} \mid \text{queue} = [(i \ominus MS)_{1}^{mo-1}, i^{mn}], sn = i \ominus MS 1, rn = i \}
\]

and

\[
\{(M_{(2a,i),(0,ao,mn,0,ret)}^{(MS, MR)}), \text{receive} \_ \text{mess} \mid \text{queue} = [i^{mn-1}], sn = i, rn = i \}
\]

which correspond exactly to the sets of arcs defined by rows 4 and 5 respectively of Table 8.3.

After substituting the restrictions on \(mo, ao, mn\) and \(ret\) from the definition of \(V_{(2a,i)}^{(MS, MR)}\), the destination markings are given by

\[
\{M_{(3a,i),(mo-1,ao,mn,0,ret)}^{(MS, MR)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR, mo \geq 1 \}
\]

and

\[
\{M_{(3a,i),(0,ao,mn-1,0,ret)}^{(MS, MR)} \mid 0 \leq ao \leq MR, 1 \leq mn \leq ret + 1, 0 \leq ret \leq MR \}
\]
respectively. From Proposition 8.1 the first set becomes

$$\{M^{(MS,MR)}_{(3a,i),(mo,ao,nn,0,ret)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR, MR \geq 1\}$$

and from Proposition 8.3 the second set becomes

$$\{M^{(MS,MR)}_{(3b,i),(0,ao,nn,0,ret)} \mid 0 \leq ao \leq MR, 0 \leq mn \leq ret, 0 \leq ret \leq MR\}$$

By inspection of row 4 of Table 8.1 the first set equals $$V^{(MS,MR)}_{(3a,i)}$$ and by inspection of row 5, the second set is a subset of $$V^{(MS,MR)}_{(3b,i)}$$.

**ack_loss:** From Lemma 6.9, all markings in $$\{M^{(MS,MR)}_{(2a,i),(mo,ao,nn,0,ret)} \mid M^{(MS,MR)}_{(2a,i),(mm,0,ret)} \in V^{(MS,MR)}_{(2a,i)}, ao \geq 1\}$$ enable **ack_loss**, with binding $$< queue = [a;ao], rn = i >$$. The resulting set of outgoing arcs is given by:

$$\{(M^{(MS,MR)}_{(2a,i),(mo,ao,nn,0,ret)}, \text{ack_loss} < queue = [a;ao], rn = i >), M^{(MS,MR)}_{(2a,i),(mo,ao-1,nn,0,ret)} \mid M^{(MS,MR)}_{(2a,i),(mo,ao,nn,0,ret)} \in V^{(MS,MR)}_{(2a,i)}, ao \geq 1, MR \geq 1\}$$

which corresponds exactly to the set of arcs defined by row 6 of Table 8.3. After substituting the restrictions on $$mo, ao, mn$$ and $$ret$$ from the definition of $$V^{(MS,MR)}_{(2a,i)}$$, the destination markings are given by

$$\{M^{(MS,MR)}_{(2a,i),(mo,ao-1,nn,0,ret)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq ret \leq MR, 0 \leq mn \leq ret + 1, ao \geq 1,$$

$$MR \geq 1\}$$

from Proposition 8.2, we have

$$\{M^{(MS,MR)}_{(2a,i),(mo,ao,nn,0,ret)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq ret \leq MR, 0 \leq mn \leq ret + 1, MR \geq 1\}$$

which, by inspection of row 2 of Table 8.1, is a subset of $$V^{(MS,MR)}_{(2a,i)}$$.

**receive_dup_ack:** From Lemma 6.7, all markings in the set $$\{M^{(MS,MR)}_{(2a,i),(mo,ao,nn,0,ret)} \mid M^{(MS,MR)}_{(2a,i),(mo,ao,nn,0,ret)} \in V^{(MS,MR)}_{(2a,i)}, ao \geq 1\}$$ also satisfy the enabling conditions of **receive_dup_ack**, with binding $$< queue = [a;ao-1], sn = i, rn = i >$$. The resulting set of outgoing arcs is given by:

$$\{(M^{(MS,MR)}_{(2a,i),(mo,ao,nn,0,ret)}, \text{receive_dup_ack} < queue = [a;ao-1], sn = i, rn = i >), M^{(MS,MR)}_{(2a,i),(mo,ao-1,nn,0,ret)} \mid M^{(MS,MR)}_{(2a,i),(mo,ao,nn,0,ret)} \in V^{(MS,MR)}_{(2a,i)}, ao \geq 1, MR \geq 1\}$$

which corresponds exactly to the set of arcs defined by row 7 of Table 8.3. After substituting the restrictions on $$mo, ao, mn$$ and $$ret$$ from the definition of $$V^{(MS,MR)}_{(2a,i)}$$, the destination markings are given by

$$\{M^{(MS,MR)}_{(2a,i),(mo,ao-1,nn,0,ret)} \mid 0 \leq mo + ao \leq MR, 0 \leq ret \leq MR, 0 \leq mn \leq ret + 1, ao \geq 1,$$

$$MR \geq 1\}$$
which is identical to the set of destination markings of the \texttt{ack\_loss} transition above. These have already been shown to be a subset of $V_{(2a,i)}^{(MS,MR)}$.

\textbf{send\_mess, send\_ack and receive\_ack:} From Lemmas 6.2, 6.5 and 6.6, none of these three transitions is enabled by any marking in $V_{(2a,i)}^{(MS,MR)}$. This corresponds to the absence of these three transitions from Table 8.3.

The binding elements enabled by the markings in $V_{(2a,i)}^{(MS,MR)}$ correspond exactly to those presented in Table 8.3. The successor marking of each enabled binding element is also a member of $V_{(2a,i)}^{(MS,MR)} \cup V_{(3a,i)}^{(MS,MR)} \cup V_{(3b,i)}^{(MS,MR)}$. Thus the sublemma is proved.

\textbf{Sublemma 8.4.3.} All enabled binding elements of all markings in $V_{(2b,i)}^{(MS,MR)}$ are described exactly by $A_{(2b,i)}^{(MS,MR)}$ in Table 8.4 and all immediate successors of all markings in $V_{(2b,i)}^{(MS,MR)}$ are contained in $V_{(2b,i)}^{(MS,MR)} \cup V_{(3b,i)}^{(MS,MR)} \cup V_{(1,i \oplus MS)^1}^{(MS,MR)}$.

\textit{Proof.} From Definition 7.6 and row 3 of Table 8.1, the markings in $V_{(2b,i)}^{(MS,MR)}$ have the form

\begin{align*}
M(\text{sender\_state}) &= 1^i \text{wait\_ack} \quad M(\text{send\_seq\_no}) = 1^i, \\
M(\text{receiver\_state}) &= 1^r \text{\_ready} \quad M(\text{recv\_seq\_no}) = 1^i \oplus MS 1, \\
M(\text{mess\_channel}) &= 1^i[i^mn] \quad M(\text{ack\_channel}) = 1^i[i^ao (i \oplus MS 1)^mn], \\
M(\text{retrans\_counter}) &= 1^i \text{ret}
\end{align*}

where $0 \leq ao \leq MR, 0 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1$ and $0 \leq ret \leq MR$. We now systematically consider the enabling and occurrence of each transition from the markings in $V_{(2b,i)}^{(MS,MR)}$:

\textbf{timeout\_retrans:} From Lemma 6.3, all markings in $\{M_{(2b,i),(0,a0,mn,an,ret)}^{(MS,MR)} | M_{(2b,i),(0,a0,mn,an,ret)}^{(MS,MR)} \in V_{(2b,i)}^{(MS,MR)}, ret < MR\}$ enable \texttt{timeout\_retrans}, with binding $<\text{queue} = [i^mn], sn = i, rc = ret >$. The resulting set of outgoing arcs is given by:

$$\{ (M_{(2b,i),(0,a0,mn,an,ret)}^{(MS,MR)}, \text{timeout\_retrans} <\text{queue} = [i^mn], sn = i, rc = ret >, \\
M_{(2b,i),(0,a0,mn+1,an,ret+1)}^{(MS,MR)} ) | M_{(2b,i),(0,a0,mn,an,ret)}^{(MS,MR)} \in V_{(2b,i)}^{(MS,MR)}, ret < MR\}$$

which corresponds exactly to the set of arcs defined by row 1 of Table 8.4. After substituting the restrictions on $ao, mn, an$ and $ret$ from the definition of $V_{(2b,i)}^{(MS,MR)}$, the destination markings are given by

$$\{ M_{(2b,i),(0,a0,mn+1,an,ret+1)}^{(MS,MR)} | 0 \leq ao \leq MR, 0 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1, \\
0 \leq ret < MR\}$$
From Proposition 8.8, we have

\[ \{ M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)} | 0 \leq ao \leq MR, 1 \leq mn \leq ret, 1 \leq mn + an \leq ret + 1, \\
1 \leq ret \leq MR \} \]

which, by inspection of row 3 of Table 8.1, this is a subset of \( V^{(MS,MR)}_{(2b,i)} \).

**mess_loss:** From Lemma 6.8, all markings in \( \{ M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)} | M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)} \in V^{(MS,MR)}_{(2b,i)}, mn \geq 1 \} \) (which implies \( ret \geq 1 \)) enable mess_loss, with binding \(< queue = [i^{mn}], sn = i >. The resulting set of outgoing arcs is given by:

\[ \{ (M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)}, mess_loss < queue = [i^{mn}], sn = i >, M^{(MS,MR)}_{(2b,i),(0,a_0,mn-1,an,ret)} | M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)} \in V^{(MS,MR)}_{(2b,i)}, mn \geq 1, ret \geq 1 \} \]

which corresponds exactly to the set of arcs defined by row 2 of Table 8.4. After substituting the restrictions on \( ao, mn, an \) and \( ret \) from the definition of \( V^{(MS,MR)}_{(2b,i)} \), the destination markings are given by

\[ \{ M^{(MS,MR)}_{(2b,i),(0,a_0,mn-1,an,ret)} | 0 \leq ao \leq MR, 1 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1, \\
0 < ret \leq MR \} \]

From Proposition 8.4, we have

\[ \{ M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)} | 0 \leq ao \leq MR, 0 \leq mn \leq ret-1, 0 \leq mn+an \leq ret, 0 < ret \leq MR \} \]

which, by inspection of row 3 of Table 8.1, is a subset of \( V^{(MS,MR)}_{(2b,i)} \).

**receive_mess:** From Lemma 6.4, all markings in \( \{ M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)} | M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)} \in V^{(MS,MR)}_{(2b,i)}, mn \geq 1 \} \) (which implies \( ret \geq 1 \)) also enable receive_mess, with binding \(< queue = [i^{mn-1}], sn = i, rn = i + MS 1 >. The resulting set of outgoing arcs is given by:

\[ \{ (M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)}, receive_mess < queue = [i^{mn-1}], sn = i, rn = i + MS 1 >, M^{(MS,MR)}_{(3b,i),(0,a_0,mn-1,an,ret)} | M^{(MS,MR)}_{(2b,i),(0,a_0,mn,an,ret)} \in V^{(MS,MR)}_{(2b,i)}, mn \geq 1, ret \geq 1 \} \]

which corresponds exactly to the set of arcs defined by row 3 of Table 8.4. After substituting the restrictions on \( ao, mn, an \) and \( ret \) from the definition of \( V^{(MS,MR)}_{(2b,i)} \), the destination markings are given by

\[ \{ M^{(MS,MR)}_{(3b,i),(0,a_0,mn-1,an,ret)} | 0 \leq ao \leq MR, 1 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1, \\
0 < ret \leq MR \} \]

From Proposition 8.4, we have

\[ \{ M^{(MS,MR)}_{(3b,i),(0,a_0,mn,an,ret)} | 0 \leq ao \leq MR, 0 \leq mn \leq ret-1, 0 \leq mn+an \leq ret, 0 < ret \leq MR \} \]

which, by inspection of row 5 of Table 8.1, is a subset of \( V^{(MS,MR)}_{(3b,i)} \).
8.5. Containment of all Successors

**ack_loss:** From Lemma 6.9, all markings in \( M_{(2b,i)}^{(MS,MR)} | M_{(2b,i)}^{(MS,MR)} \in V_{(2b,i)}^{(MS,MR)}, ao + an \geq 1 \) satisfy the enabling conditions of **ack_loss**, with binding \( <queue = [i^{ao} (i + MS 1)^{an}], rn = i > \) provided \( ao \geq 1 \) (implying \( MR \geq 1 \)) or \( <queue = [i^{ao} (i + MS 1)^{an}], rn = i + MS 1 \) > provided \( an \geq 1 \). The resulting sets of outgoing arcs are given by:

\[
(M_{(2b,i)}^{(MS,MR)},M_{(2b,i)}^{(MS,MR)}) \cdot \text{ack_loss} <\text{queue} = [i^{ao} (i + MS 1)^{an}], rn = i >,
\]

\[
M_{(2b,i)}^{(MS,MR)} | M_{(2b,i)}^{(MS,MR)} \in V_{(2b,i)}^{(MS,MR)}, ao \geq 1, MR \geq 1
\]

and

\[
(M_{(2b,i)}^{(MS,MR)},M_{(2b,i)}^{(MS,MR)}) \cdot \text{ack_loss} <\text{queue} = [i^{ao} (i + MS 1)^{an}], rn = i + MS 1 >,
\]

\[
M_{(2b,i)}^{(MS,MR)} | M_{(2b,i)}^{(MS,MR)} \in V_{(2b,i)}^{(MS,MR)}, an \geq 1
\]

which correspond exactly to the sets of arcs defined by rows 4 and 5 respectively of Table 8.4.

After substituting the restrictions on \( ao, mn, an \) and \( ret \) from the definition of \( V_{(2b,i)}^{(MS,MR)} \), the destination markings are given by

\[
\{ M_{(2b,i)}^{(MS,MR)} | 1 \leq ao \leq MR, 0 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1,
\]

\[
0 \leq ret \leq MR, MR \geq 1
\]

and

\[
\{ M_{(2b,i)}^{(MS,MR)} | 0 \leq ao \leq MR, 0 \leq mn \leq ret,
\]

\[
0 \leq mn + an \leq ret + 1, 0 \leq ret \leq MR, an \geq 1
\]

respectively. From Proposition 8.2 the first set of markings becomes

\[
\{ M_{(2b,i)}^{(MS,MR)} | 0 \leq ao \leq MR - 1, 0 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1,
\]

\[
0 \leq ret \leq MR, MR \geq 1
\]

and from Proposition 8.5 the second set of markings becomes

\[
\{ M_{(2b,i)}^{(MS,MR)} | 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR
\]

By inspection of row 3 of Table 8.1, both sets are subsets of \( V_{(2b,i)}^{(MS,MR)} \).

**receive_dup_ack:** From Lemma 6.7, all markings in \( \{ M_{(2b,i)}^{(MS,MR)} | M_{(2b,i)}^{(MS,MR)} \in V_{(2b,i)}^{(MS,MR)}, ao \geq 1, MR \geq 1 \} \) also enable **receive_dup_ack**, with binding \( <queue = [i^{ao-1} (i + MS 1)^{an}], sn = i, rn = i > \). The resulting set of outgoing arcs is given by:

\[
(M_{(2b,i)}^{(MS,MR)},M_{(2b,i)}^{(MS,MR)}) \cdot \text{receive_dup_ack} <\text{queue} = [i^{ao-1} (i + MS 1)^{an}], sn = i, rn = i >,
\]

\[
M_{(2b,i)}^{(MS,MR)} | M_{(2b,i)}^{(MS,MR)} \in V_{(2b,i)}^{(MS,MR)}, ao \geq 1, MR \geq 1
\]
which corresponds exactly to the set of arcs defined by row 6 of Table 8.4. After substituting the restrictions on \( ao, mn, an \) and \( ret \) from the definition of \( V^{(MS,MR)}_{(2b,i)} \), the destination markings are given by

\[
\{ M^{(MS,MR)}_{(2b,i),(0,ao-1,mm,an,ret)} \mid 1 \leq ao \leq MR, 0 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1, \\
0 \leq ret \leq MR, MR \geq 1 \}
\]

which is identical to the set of destination markings of the \textbf{ack\_loss} transition above, when it is losing old acknowledgements. These have already been shown to be a subset of \( V^{(MS,MR)}_{(2b,i)} \).

**receive\_ack:** From Lemma 6.6, all markings in \( V^{(MS,MR)}_{(2b,i)} \) enable \textbf{receive\_ack}, with binding \( < queue \) = \([(i \oplus MS 1)^{(an-1)}, sn = i, rn = i \oplus MS 1, rc = ret > \). The resulting set of outgoing arcs is given by:

\[
\{ (M^{(MS,MR)}_{(2b,i),(0,ao,mm,an,ret)}, \textbf{receive\_ack} < queue \) \mid (i \oplus MS 1)^{(an-1)}, sn = i, rn = i \oplus MS 1, \\
rc = ret > , M^{(MS,MR)}_{(1,i\oplus MS 1),(mn,an-1,0,0,0)} \mid M^{(MS,MR)}_{(2b,i),(0,ao,mm,an,ret)} \in V^{(MS,MR)}_{(2b,i)}, an \geq 1 \}
\]

which corresponds exactly to the set of arcs defined by row 7 of Table 8.4. After substituting the relevant restrictions on \( ao, mn, an \) and \( ret \) from the definition of \( V^{(MS,MR)}_{(2b,i)} \), the destination markings are given by

\[
\{ M^{(MS,MR)}_{(1,i\oplus MS 1),(mn,an-1,0,0,0)} \mid 0 \leq ret \leq MR, 0 \leq mn \leq ret, 0 \leq mn + an \leq ret + 1, an \geq 1 \}
\]

If \( 0 \leq ret \leq MR \) and \( 0 \leq mn \leq ret \), then \( 0 \leq mn \leq MR \). Similarly, \( 0 \leq mn + an \leq MR+1 \). Substituting these inequalities gives us

\[
\{ M^{(MS,MR)}_{(1,i\oplus MS 1),(mn,an-1,0,0,0)} \mid 0 \leq mn \leq MR, 0 \leq mn + an \leq MR + 1, an \geq 1 \}
\]

We can now follow a procedure very similar to that in Proposition 8.5, except that \( MR \) replaces \( ret \) and the new messages and new acknowledgements are now considered to be old messages and old acknowledgements, due to the sender sequence number having increased from \( i \) to \( i \oplus MS 1 \). The result is

\[
\{ M^{(MS,MR)}_{(1,i\oplus MS 1),(mn,an,0,0,0)} \mid 0 \leq mn \leq MR, 0 \leq mn + an \leq MR \}
\]

which equals \( V^{(MS,MR)}_{(1,i\oplus MS 1)} \) from Table 8.1, with \( i \oplus MS 1 \) substituted for \( i \).

**send\_mess** and **send\_ack:** From Lemmas 6.2 and 6.5, neither of these transitions is enabled by any marking in \( V^{(MS,MR)}_{(2b,i)} \). This corresponds to the absence of these two transitions from Table 8.4.

The binding elements enabled by the markings in \( V^{(MS,MR)}_{(2b,i)} \) correspond exactly to those presented in Table 8.4 and the successor marking of each enabled binding element is also a member of \( V^{(MS,MR)}_{(2b,i)} \cup V^{(MS,MR)}_{(3b,i)} \cup V^{(MS,MR)}_{(1,i\oplus MS 1)} \). Thus the sublemma is proved.

\[ \square \]
8.5. Containment of all Successors

Sublemma 8.4.4. All enabled binding elements of all markings in \( V_{(3a,i)}^{(MS,MR)} \) are described exactly by \( A_{(3a,i)}^{(MS,MR)} \) in Table 8.5 and all immediate successors of all markings in \( V_{(3a,i)}^{(MS,MR)} \) are contained in \( V_{(3a,i)}^{(MS,MR)} \) where, for a marking \( M \in V_{(3a,i)}^{(MS,MR)} \):

\[
\begin{align*}
    M(\text{sender\_state}) & = 1^i \text{wait\_ack} & M(\text{send\_seq\_no}) & = 1^i \\
    M(\text{receiver\_state}) & = 1^i \text{process} & M(\text{recv\_seq\_no}) & = 1^i \\
    M(\text{mess\_channel}) & = 1^i \lfloor (i \ominus MS 1)^{mo} i^{mn} \rfloor & M(\text{ack\_channel}) & = 1^i \lfloor a^{ao} \rfloor \\
    M(\text{retrans\_counter}) & = 1^i \text{ret}
\end{align*}
\]

where \( 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq ret + 1 \) and \( 0 \leq ret \leq MR \). We now systematically consider enabling and occurrence of each transition from the markings in \( V_{(3a,i)}^{(MS,MR)} \):

**timeout_retrans:** From Lemma 6.3, all markings in \( \{ M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \in V_{(3a,i)}^{(MS,MR)}, ret < MR \} \) enable \text{timeout\_retrans}, with binding \( \langle \text{queue} = [(i \ominus MS 1)^{mo} i^{mn}], sn = i, rc = ret > \). The resulting set of outgoing arcs is given by:

\[
\begin{align*}
    \{ (M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}), & < \text{queue} = [(i \ominus MS 1)^{mo} i^{mn}], sn = i, rc = ret >, \\
    M_{(3a,i),(mo,ao,mn+1,ret+1)}^{(MS,MR)} ) & \mid M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \in V_{(3a,i)}^{(MS,MR)}, ret < MR \}
\end{align*}
\]

which corresponds exactly to the set of arcs defined by row 1 of Table 8.5. After substituting the restrictions on \( mo, ao, mn \) and \( ret \) from the definition of \( V_{(3a,i)}^{(MS,MR)} \), the destination markings are given by

\[
\{ M_{(3a,i),(mo,ao,mn+1,ret+1)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq ret + 1, 0 \leq ret < MR \}
\]

From Proposition 8.7, we have

\[
\{ M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid 0 \leq mo + ao \leq MR - 1, 1 \leq mn \leq ret + 1, 1 \leq ret < MR \}
\]

which, by inspection of row 4 of Table 8.1, is a subset of \( V_{(3a,i)}^{(MS,MR)} \).

**mess Loss:** From Lemma 6.8, all markings in \( \{ M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \mid M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \in V_{(3a,i)}^{(MS,MR)}, mo + mn \geq 1 \} \) enable \text{mess\_loss}, with binding \( \langle \text{queue} = [(i \ominus MS 1)^{mo} i^{mn}], sn = i \ominus MS 1 > \) provided \( mo \geq 1 \) (implying \( MR - 1 \geq 1 \)) or \( \langle \text{queue} = [(i \ominus MS 1)^{mo} i^{mn}], sn = i > \) provided \( mn \geq 1 \). The resulting sets of outgoing markings are given by:

\[
\begin{align*}
    \{ (M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)}), & < \text{queue} = [(i \ominus MS 1)^{mo} i^{mn}], sn = i \ominus MS 1 >, \\
    M_{(3a,i),(mo-1,ao,mn,0,ret)}^{(MS,MR)} ) & \mid M_{(3a,i),(mo,ao,mn,0,ret)}^{(MS,MR)} \in V_{(3a,i)}^{(MS,MR)}, mo \geq 1, MR - 1 \geq 1 \}
\end{align*}
\]

145
M(MS, MR) \mid (3a, i), (mo, ao, mn, 0, ret) \in V(MS, MR)_{(3a, i)}$ and restrictions on $V(MS, MR)_{(3a, i)}$, respectively. Replacing $MR$ with $MR - 1$ in Proposition 8.1 gives us

$$M(MS, MR)_{(3a, i), (mo, ao, mn, 0, ret)} \mid 0 \leq mo + ao \leq MR - 2, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR,$$

$$MR - 1 \geq 1} \}

By inspection of row 4 of Table 8.1, both sets of destination markings are subsets of $V(MS, MR)_{(3a, i)}$. For the first set of markings, and from Proposition 8.3 the second set of markings becomes

$$M(MS, MR)_{(3a, i), (mo, ao, mn, 0, ret)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR$$

ack_loss: From Lemma 6.9, all markings in $\{M(MS, MR)_{(3a, i), (mo, ao, mn, 0, ret)} \mid M(MS, MR)_{(3a, i), (mo, ao, mn, 0, ret)} \in V(MS, MR)_{(3a, i)}, ao \geq 1\}$ (implying $MR - 1 \geq 1$) enable ack_loss, with binding $<queue = [i^{ao}], rn = i >$. The resulting set of outgoing arcs is given by:

$$\{((M(MS, MR)_{(3a, i), (mo, ao, mn, 0, ret)}, ack_loss <queue = [i^{ao}], rn = i >, M(MS, MR)_{(3a, i), (mo, ao, mn, 0, ret)}) \mid M(MS, MR)_{(3a, i), (mo, ao, mn, 0, ret)} \in V(MS, MR)_{(3a, i)}, ao \geq 1, MR - 1 \geq 1\}$$

which corresponds exactly to the set of arcs defined by row 4 of Table 8.5. After substituting the restrictions on $mo, ao, mn$ and $ret$ from the definition of $V(MS, MR)_{(3a, i)}$, the destination markings are given by

$$\{M(MS, MR)_{(3a, i), (mo, ao, mn, 0, ret)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR,$$

$$ao \geq 1, MR - 1 \geq 1\} \}

Replacing $MR$ with $MR - 1$ in Proposition 8.2, we get

$$\{M(MS, MR)_{(3a, i), (mo, ao, mn, 0, ret)} \mid 0 \leq mo + ao \leq MR - 2, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR,$$

$$MR - 1 \geq 1\} \}

which, by inspection of row 4 of Table 8.1, is a subset of $V(MS, MR)_{(3a, i)}$. 

146
8.5. Containment of all Successors

receive_dup_ack: From Lemma 6.7, all markings in \( \{ M_{(3a,i)}^{(MS,MR)} | M_{(3a,i)}^{(MS,MR)} \in V_{(3a,i)}^{(MS,MR)} \} \) (implying \( MR = 1 \geq 1 \)) also enable receive_dup_ack, with binding \(< queue = [i^{ao-1}], sn = i, rn = i >\). The resulting set of outgoing arcs is given by:

\[
\{ (M_{(3a,i)}^{(MS,MR)}), receive\_dup\_ack < queue = [i^{ao-1}], sn = i, rn = i >, M_{(3a,i)}^{(MS,MR)} | M_{(3a,i)}^{(MS,MR)} \in V_{(3a,i)}^{(MS,MR)}, ao \geq 1, MR = 1 \geq 1 \}
\]

which corresponds exactly to the set of arcs defined by row 5 of Table 8.5. After substituting the restrictions on \( mo, ao, mn \) and \( ret \) from the definition of \( V_{(3a,i)}^{(MS,MR)} \), the destination markings are given by

\[
\{ M_{(3a,i)}^{(MS,MR)} | 0 \leq ao + mo \leq MR - 1, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR, ao \geq 1, MR - 1 \geq 1 \}
\]

which is identical to the set of destination markings of the \texttt{ack} transition above. These have already been shown to be a subset of \( V_{(3a,i)}^{(MS,MR)} \).

send_ack: From Lemma 6.5, all markings in \( V_{(3a,i)}^{(MS,MR)} \) enable send_ack, with binding \(< queue = [i^{ao}], sn = i >\). The resulting set of outgoing arcs is given by:

\[
\{ (M_{(3a,i)}^{(MS,MR)}), send\_ack < queue = [i^{ao}], sn = i >, M_{(3a,i)}^{(MS,MR)} | M_{(2a,i)}^{(MS,MR)} \in V_{(3a,i)}^{(MS,MR)} \}
\]

which corresponds exactly to the set of arcs defined by row 6 of Table 8.5. After substituting the restrictions on \( mo, ao, mn \) and \( ret \) from the definition of \( V_{(3a,i)}^{(MS,MR)} \), the destination markings are given by

\[
\{ M_{(2a,i)}^{(MS,MR)} | 0 \leq ao + mo \leq MR - 1, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR \}
\]

Adding 1 to each term in \( 0 \leq ao + mo \leq MR - 1 \) gives \( 1 \leq ao + mo + 1 \leq MR \). Note however that the range of \( mo \) is unaffected. This can be substituted into the set of destination markings, along with \( ao' = ao + 1 \), to get

\[
\{ M_{(2a,i)}^{(MS,MR)} | 0 \leq ao + mo \leq MR - 1, 1 \leq ao' \leq MR, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR \}
\]

Dropping the prime notation gives

\[
\{ M_{(2a,i)}^{(MS,MR)} | 0 \leq ao + mo \leq MR - 1, 1 \leq ao \leq MR, 0 \leq mn \leq ret + 1, 0 \leq ret \leq MR \}
\]

which, by inspection of row 2 of Table 8.1, is a subset of \( V_{(2a,i)}^{(MS,MR)} \).
send\_mess, receive\_mess and receive\_ack: From Lemmas 6.2, 6.4 and 6.6, none of these three transitions is enabled by any marking in $V_{(3a,i)}^{(MS,MR)}$. This corresponds to the absence of these three transitions from Table 8.5.

The binding elements enabled by the markings in $V_{(3a,i)}^{(MS,MR)}$ correspond exactly to those presented in Table 8.5 and the successor marking of each enabled binding element is also a member of $V_{(2a,i)}^{(MS,MR)}$. Thus the sublemma is proved.

Sublemma 8.4.5. All enabled binding elements of all markings in $V_{(3b,i)}^{(MS,MR)}$ are described exactly by $M_{(3b,i)}^{(MS,MR)}$ in Table 8.6 and all immediate successors of all markings in $V_{(3b,i)}^{(MS,MR)}$ are contained in $V_{(3a,i)}^{(MS,MR)}$. This corresponds to the absence of these three:

\[ M_{(3b,i)}^{(MS,MR)} \subseteq \{ (M_{(3b,i)}^{(MS,MR)}, (0,ao,mn,an,ret), \text{timeout\_retrans}) \mid \text{timeout\_retrans < queue = } [i^{mn}], sn = i, rc = ret \} \]

Proof. From Definition 7.6 and row 5 of Table 8.1, the markings in $V_{(3b,i)}^{(MS,MR)}$ have the form $M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)}$ where, for a marking $M \in V_{(4,i)}^{(MS,MR)}$:

\[
M(\text{sender\_state}) = 1^i \text{wait\_ack} \quad M(\text{send\_seq\_no}) = 1^i
\]

\[
M(\text{receiver\_state}) = 1^i \text{process} \quad M(\text{recv\_seq\_no}) = 1^i \oplus_{MS} 1
\]

\[
M(\text{mess\_channel}) = 1^i [i^{mn}] \quad M(\text{ack\_channel}) = 1^i [i^{ao} (i \oplus_{MS} 1)^{an}]
\]

\[
M(\text{retrans\_counter}) = 1^i \text{ret}
\]

where $0 \leq ao \leq MR, 0 \leq mn + an \leq ret$ and $0 \leq ret \leq MR$. We now systematically consider the enabling and occurrence of each transition from the markings in $V_{(3b,i)}^{(MS,MR)}$:

timeout\_retrans: From Lemma 6.3, all markings in $\{ M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \mid M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \in V_{(3b,i)}^{(MS,MR)} \text{, ret < MR} \}$ enable timeout\_retrans, with binding $\text{< queue = } [i^{mn}], sn = i, rc = ret$.

The resulting set of outgoing arcs is given by:

\[
\{ (M_{(3b,i),(0,ao,mn,an,ret + 1)}^{(MS,MR)}, \text{timeout\_retrans} < \text{queue = } [i^{mn}], sn = i, rc = ret) \} = M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \cap V_{(3b,i)}^{(MS,MR)} \}

which corresponds exactly to the set of arcs defined by row 1 of Table 8.6. After substituting the restrictions on $ao, mn, an$ and $ret$ from the definition of $V_{(3b,i)}^{(MS,MR)}$, the destination markings are given by

\[
\{ M_{(3b,i),(0,ao,mn,an,ret + 1)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret < MR \}
\]

From (a slight variation of) Proposition 8.7, we have

\[
\{ M_{(3b,i),(0,ao,mn,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 1 \leq ret \leq MR \}
\]

which, by inspection of row 5 of Table 8.1, is a subset of $V_{(3b,i)}^{(MS,MR)}$. 

148
8.5. Containment of all Successors

**mess** \_\_loss: From Lemma 6.8, all markings in \( \{ M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \mid M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \in V^{(MS,MR)}_{(3b,i)}, mn \geq 1 \} \) (implying \( ret \geq 1 \)) satisfy the enabling conditions of **mess** \_\_loss, with binding \( <queue = [i^{mn}], sn = i> \). The resulting set of outgoing arcs is given by:

\[
\{ (M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \cdot **mess** \_\_loss < queue = [i^{mn}], sn = i >, \\
M^{(MS,MR)}_{(3b,i),(0,ao,mn-1,an,ret)} \mid M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \in V^{(MS,MR)}_{(3b,i)}, mn \geq 1, ret \geq 1 \}
\]

which corresponds exactly to the sets of arcs defined by row 2 of Table 8.6. After substituting the restrictions on \( ao, mn, an \) and \( ret \) from the definition of \( V^{(MS,MR)}_{(3b,i)} \), the destination markings are given by

\[
\{ M^{(MS,MR)}_{(3b,i),(0,ao,mn-1,an,ret)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 1 \leq ret \leq MR, mn \geq 1 \}
\]

From (a slight variant of) Proposition 8.4, we have

\[
\{ M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret - 1, 1 \leq ret \leq MR \}
\]

which, by inspection of row 5 of Table 8.1, is a subset of \( V^{(MS,MR)}_{(3b,i)} \).

**ack** \_\_loss: From Lemma 6.9, all markings in \( \{ M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \mid M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \in V^{(MS,MR)}_{(3b,i)}, ao + an \geq 1 \} \) enable **ack** \_\_loss, with binding \( <queue = [i^{ao} (i \oplus MS 1)^{an}], rn = i > \) provided \( ao \geq 1 \) (implying \( MR \geq 1 \)) or \( <queue = [i^{ao} (i \oplus MS 1)^{an}], rn = i \oplus MS 1 > \) provided \( an \geq 1 \) (implying \( ret \geq 1 \)). The resulting sets of outgoing arcs are given by:

\[
\{ (M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \cdot **ack** \_\_loss < queue = [i^{ao} (i \oplus MS 1)^{an}], sn = i >, \\
M^{(MS,MR)}_{(3b,i),(0,ao-1,mn,an,ret)} \mid M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \in V^{(MS,MR)}_{(3b,i)}, ao \geq 1, MR \geq 1 \}
\]

and

\[
\{ (M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \cdot **ack** \_\_loss < queue = [i^{ao} (i \oplus MS 1)^{an}], sn = i \oplus MS 1 >, \\
M^{(MS,MR)}_{(3b,i),(0,ao,mn-1,an,ret)} \mid M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \in V^{(MS,MR)}_{(3b,i)}, an \geq 1, ret \geq 1 \}
\]

which corresponds exactly to the sets of arcs defined by rows 3 and 4 respectively of Table 8.6. After substituting the restrictions on \( ao, mn, an \) and \( ret \) from the definition of \( V^{(MS,MR)}_{(3b,i)} \), the destination markings are given by

\[
\{ M^{(MS,MR)}_{(3b,i),(0,ao-1,mn,an,ret)} \mid 1 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR, MR \geq 1 \}
\]

and

\[
\{ M^{(MS,MR)}_{(3b,i),(0,ao,mn,an-1,ret)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 1 \leq ret \leq MR, an \geq 1 \}
\]

respectively. From Proposition 8.2 the first set becomes

\[
\{ M^{(MS,MR)}_{(3b,i),(0,ao,mn,an,ret)} \mid 0 \leq ao \leq MR - 1, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR, MR \geq 1 \}
\]

149
and from Proposition 8.6 the second set becomes

\[ \{ M^{(MS,MR)}_{(3b,1), (0, ao, mn, an, ret)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret - 1, 1 \leq ret \leq MR \} \]

By inspection of row 5 of Table 8.1, both sets are subsets of \( V^{(MS,MR)}_{(3b,1)} \).

**receive_dup_ack**: From Lemma 6.7, all markings in \( M^{(MS,MR)}_{(3b,1), (0, ao, mn, an, ret)} \in V^{(MS,MR)}_{(3b,1)} \), \( ao \geq 1, MR \geq 1 \) also enable **receive_dup_ack**, with binding \( <queue = [^{ao-1} (i \oplus MS 1)^{an}], sn = i, rn = i> \). The resulting set of outgoing arcs is given by:

\[ \{ (M^{(MS,MR)}_{(3b,1), (0, ao, mn, an, ret)}, receive_dup_ack <queue = [^{ao-1} (i \oplus MS 1)^{an}], sn = i, rn = i>, M^{(MS,MR)}_{(3b,1), (0, ao, mn, an, ret)}) \mid M^{(MS,MR)}_{(3b,1), (0, ao, mn, an, ret)} \in V^{(MS,MR)}_{(3b,1)}, ao \geq 1, MR \geq 1 \} \]

which corresponds exactly to the set of outgoing arcs defined by row 5 of Table 8.6. After substituting the restrictions on \( ao, mn, an \) and \( ret \) from the definition of \( V^{(MS,MR)}_{(3b,1)} \), the destination markings are given by

\[ \{ M^{(MS,MR)}_{(3b,1), (0, ao-1, mn, an, ret)} \mid 1 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR, MR \geq 1 \} \]

which is identical to the set of destination markings of the **ack_loss** transition above, when losing old acknowledgements. These have already been shown to be a subset of \( V^{(MS,MR)}_{(3b,1)} \).

**receive_ack**: From Lemma 6.6, all markings in \( M^{(MS,MR)}_{(3b,1), (0, 0, 0, mn, ret)} \in V^{(MS,MR)}_{(3b,1)} \), \( an \geq 1 \) (implying \( ret \geq 1 \)) enable **receive_ack**, with binding \( <queue = [(i \oplus MS 1)^{an-1}], sn = i, rn = i \oplus MS 1, rc = ret> \). The resulting set of outgoing arcs is given by:

\[ \{ (M^{(MS,MR)}_{(3b,1), (0, 0, mn, an, ret)}, receive_ack <queue = [(i \oplus MS 1)^{an-1}], sn = i, rn = i \oplus MS 1, rc = ret>, M^{(MS,MR)}_{(3b,1), (0, 0, mn, an, ret)}) \mid M^{(MS,MR)}_{(3b,1), (0, 0, mn, an, ret)} \in V^{(MS,MR)}_{(3b,1)}, an \geq 1, ret \geq 1 \} \]

which corresponds exactly to the set of arcs defined by row 6 of Table 8.6. After substituting the restrictions on \( mn, an \) and \( ret \) from the definition of \( V^{(MS,MR)}_{(3b,1)} \) (recall that \( ao = 0 \)), the destination markings are given by

\[ \{ M^{(MS,MR)}_{(4, i \oplus MS 1), (mn, an-1, 0, 0, 0)} \mid 0 \leq mn + an \leq ret, 1 \leq ret \leq MR, an \geq 1 \} \]

If \( 1 \leq ret \leq MR \) and \( 0 \leq mn + an \leq ret \) but \( 1 \leq an \), then \( 1 \leq mn + an \leq MR \) (or, indeed, \( 1 \leq mn + an \leq MR \) combined with \( an \geq 1 \)) and \( MR \geq 1 \). This simplification gives us

\[ \{ M^{(MS,MR)}_{(4, i \oplus MS 1), (mn, an-1, 0, 0, 0)} \mid 0 \leq mn + an \leq MR, an \geq 1, MR \geq 1 \} \]

Replacing \( ret \) with \( MR \) in Proposition 8.6, and considering the new messages and new acknowledgements as old messages and old acknowledgements (due to the sender sequence number having incremented) we get

\[ \{ M^{(MS,MR)}_{(4, i \oplus MS 1), (mn, an, 0, 0, 0)} \mid 0 \leq mn + an \leq MR - 1, MR \geq 1 \} \]
which equals $V^{(MS,MR)}_{\langle 4,2,MS_1 \rangle}$ from row 6 of Table 8.1, with $i \oplus MS_1$ substituted for $i$.

**send_ack:** From Lemma 6.5, all markings in $V^{(MS,MR)}_{\langle 3b,i \rangle}$ enable send_ack, with binding $< queue = [i^{ao} (i \oplus MS_1)^{an}], rn = i \oplus MS_1 >$. The resulting set of outgoing arcs is given by:

$$\{ M^{(MS,MR)}_{\langle 3b,i,0,ao,nn,an,ret \rangle}, send_ack < queue = [i^{ao} (i \oplus MS_1)^{an}], rn = i \oplus MS_1 >, M^{(MS,MR)}_{\langle 2b,i,0,ao,nn,an+1,ret \rangle} \mid M^{(MS,MR)}_{\langle 3b,i,0,ao,nn,an,ret \rangle} \in V^{(MS,MR)}_{\langle 3b,i \rangle} \}$$

which corresponds exactly to the set of markings defined by row 7 of Table 8.6. After substituting the restrictions on $ao$, $nn$, $an$ and $ret$ from the definition of $V^{(MS,MR)}_{\langle 3b,i \rangle}$, the destination markings are given by

$$\{ M^{(MS,MR)}_{\langle 2b,i,0,ao,nn,an+1,ret \rangle} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR \}$$

Adding 1 to each term in $0 \leq mn + an \leq ret$ gives $1 \leq mn + an + 1 \leq ret + 1$. However, note that the range of $mn$ is not affected. Thus, this set can be expressed as

$$\{ M^{(MS,MR)}_{\langle 2b,i,0,ao,nn,an+1,ret \rangle} \mid 0 \leq ao \leq MR, 0 \leq mn \leq ret, 1 \leq mn + an + 1 \leq ret + 1, 0 \leq ret \leq MR \}$$

Substituting $an' = an + 1$ and then eliminating the prime from $an'$ results in

$$\{ M^{(MS,MR)}_{\langle 2b,i,0,ao,nn,an,ret \rangle} \mid 0 \leq ao \leq MR, 0 \leq mn - ret, 1 \leq mn + an \leq ret + 1, 0 \leq ret \leq MR \}$$

which, by inspection of row 3 of Table 8.1, is a subset of $V^{(MS,MR)}_{\langle 2b,i \rangle}$.

**send_mess and receive_mess:** From Lemmas 6.2 and 6.4, neither of these transitions is enabled by any marking in $V^{(MS,MR)}_{\langle 3b,i \rangle}$. This corresponds to the absence of these two transitions from Table 8.6.

The binding elements enabled by the markings in $V^{(MS,MR)}_{\langle 3b,i \rangle}$ correspond exactly to those presented in Table 8.6 and the successor marking of each enabled binding element is also a member of $V^{(MS,MR)}_{\langle 2b,i \rangle} \cup V^{(MS,MR)}_{\langle 3b,i \rangle} \cup V^{(MS,MR)}_{\langle 4,2,MS_1 \rangle}$. Thus the sublemma is proved.

**Sublemma 8.4.6.** All enabled binding elements of all markings in $V^{(MS,MR)}_{\langle 4,i \rangle}$ are described exactly by $M^{(MS,MR)}_{\langle 4,i \rangle}$ in Table 8.7 and all immediate successors of all markings in $V^{(MS,MR)}_{\langle 4,i \rangle}$ are contained in $V^{(MS,MR)}_{\langle 3a,i \rangle} \cup V^{(MS,MR)}_{\langle 4,i \rangle} \cup V^{(MS,MR)}_{\langle 1,i \rangle}$.

**Proof.** From Definition 7.6 and row 6 of Table 8.1, the markings in $V^{(MS,MR)}_{\langle 4,i \rangle}$ have the form $M^{(MS,MR)}_{\langle 4,i,0,0,0,0 \rangle}$ where, for a marking $M \in V^{(MS,MR)}_{\langle 4,i \rangle}$:

- $M(sender_state) = 1's_{ready}$
- $M(receiver_state) = 1's_{process}$
- $M(mess\_channel) = 1's_{i \oplus MS_1}^{mo}$
- $M(ack\_channel) = 1's_{i^{ao}}$
- $M(retrans\_counter) = 1's_0$
where $0 \leq mo + ao \leq MR - 1$ and $MR \geq 1$. We now systematically consider the enabling and occurrence of each transition from the markings in $V^{(MS,MR)}_{(4,i)}$.

**send\_mess:** From Lemma 6.2, all markings in $V^{(MS,MR)}_{(4,i)}$ enable send\_mess, with binding $<queue = [(i \odot MS 1)^{mo}], sn = i>$. The resulting set of outgoing arcs is given by:

$$\{(M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)}, \text{send\_mess} <queue = [(i \odot MS 1)^{mo}], sn = i >, M^{(MS,MR)}_{(3a,i),(mo,ao,1,0,0)}, M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(4,i)}\}$$

which corresponds exactly to the set of arcs defined by row 1 of Table 8.7. After substituting the restrictions on $mo$ and $ao$ from the definition of $V^{(MS,MR)}_{(4,i)}$, the destination markings are given by $\{M^{(MS,MR)}_{(3a,i),(mo,ao,1,0,0)} | 0 \leq mo + ao \leq MR - 1\}$. By inspection of row 4 of Table 8.1, this is a subset of $V^{(MS,MR)}_{(3a,i)}$.

**mess\_loss:** From Lemma 6.8, all markings in $\{M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} | M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(4,i)}, mo \geq 1\}$ (implying $MR - 1 \geq 1$) enables mess\_loss, with binding $<queue = [i \odot MS 1^{mo}], sn = i \odot MS 1 >$. The resulting set of outgoing arcs is given by:

$$\{(M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)}, \text{mess\_loss} <queue = [i \odot MS 1^{mo}], sn = i \odot MS 1 >, M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)}, M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(4,i)}, mo \geq 1, MR - 1 \geq 1\}$$

which corresponds exactly to the set of arcs defined by row 2 of Table 8.7. After substituting the restrictions on $mo$ and $ao$ from the definition of $V^{(MS,MR)}_{(4,i)}$, the destination markings are given by $\{M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} | 0 \leq mo + ao \leq MR - 1, mo \geq 1, MR - 1 \geq 1\}$. From Proposition 8.1 we have $\{M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} | 0 \leq mo + ao \leq MR - 2, MR - 1 \geq 1\}$, which, by inspection of row 6 of Table 8.1, is a subset of $V^{(MS,MR)}_{(4,i)}$.

**ack\_loss:** From Lemma 6.9, all markings in $\{M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} | M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(4,i)}, ao \geq 1\}$ (implying $MR - 1 \geq 1$) enable ack\_loss, with binding $<queue = [i^{ao}], rn = i >$. The resulting set of outgoing arcs is given by:

$$\{(M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)}, \text{ack\_loss} <queue = [i^{ao}], rn = i >, M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)}, M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(4,i)}, ao \geq 1, MR - 1 \geq 1\}$$

which corresponds exactly to the set of arcs defined by row 3 of Table 8.7. After substituting the restrictions on $mo$ and $ao$ from the definition of $V^{(MS,MR)}_{(4,i)}$, the destination markings are given by $\{M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} | 0 \leq mo + ao \leq MR - 1, ao \geq 1, MR - 2 \geq 1\}$. From Proposition 8.2 we have $\{M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} | 0 \leq mo + ao \leq MR - 2, ao \geq 1, MR - 1 \geq 1\}$ which, by inspection of row 6 of Table 8.1, is a subset of $V^{(MS,MR)}_{(4,i)}$.

152
receive\_dup\_ack: From Lemma 6.7, all markings in \( \{ M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \mid M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(4,i)}, ao \geq 1, MR - 1 \geq 1 \} \) also enable receive\_dup\_ack, with binding \( \langle \text{queue} = [i^{ao-1}], sn = i, rn = i \rangle \). The resulting set of outgoing arcs is given by:

\[
\{ (M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \cdot \text{receive\_dup\_ack} \langle \text{queue} = [i^{ao-1}], sn = i, rn = i \rangle, M^{(MS,MR)}_{(4,i),(mo,ao-1,0,0,0)}) \mid M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(4,i)}, ao \geq 1 \}
\]

which corresponds exactly to the set of arcs defined by row 4 of Table 8.7. The set of destination markings is identical to that resulting from occurrence of \text{ack\_loss} above, which has already been shown to be a subset of \( V^{(MS,MR)}_{(4,i)} \).

send\_ack: From Lemma 6.5, all markings in \( V^{(MS,MR)}_{(4,i)} \) satisfy the enabling conditions of send\_ack, with binding \( \langle \text{queue} = [i^{ao}], rn = i \rangle \). The resulting set of outgoing arcs is given by:

\[
\{ (M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \cdot \text{send\_ack} \langle \text{queue} = [i^{ao}], rn = i \rangle, M^{(MS,MR)}_{(4,i),(mo,ao+1,0,0,0)}) \mid M^{(MS,MR)}_{(4,i),(mo,ao,0,0,0)} \in V^{(MS,MR)}_{(4,i)} \}
\]

which corresponds exactly to the set of arcs defined by row 5 of Table 8.7. After substituting the restrictions on \text{mo} and \text{ao} from the definition of \( V^{(MS,MR)}_{(4,i)} \), the destination markings are given by

\[
\{ M^{(MS,MR)}_{(1,i),(mo,ao+1,0,0,0)} \mid 0 \leq mo + ao \leq MR - 1 \}.
\]

Adding 1 to each term in \( 0 \leq mo + ao \leq MR - 1 \) gives \( 1 \leq mo + ao \leq MR \). However, note that the range of \text{mo} is unaffected. Thus this set can be written as

\[
\{ M^{(MS,MR)}_{(1,i),(mo,ao+1,0,0,0)} \mid 0 \leq mo \leq MR - 1, 1 \leq mo + ao + 1 \leq MR \}
\]

Substituting \( ao' = ao + 1 \) and then removing the prime notation gives

\[
\{ M^{(MS,MR)}_{(1,i),(mo,ao,0,0,0)} \mid 0 \leq mo \leq MR - 1, 1 \leq mo + ao \leq MR \}
\]

which, by inspection of row 1 of Table 8.1, is a subset of \( V^{(MS,MR)}_{(1,i)} \).

timeout\_retrans, receive\_mess and receive\_ack: From Lemmas 6.3, 6.4 and 6.6 none of these transitions is enabled by any marking in \( V^{(MS,MR)}_{(4,i)} \). This corresponds to the absence of these three transitions from Table 8.7.

The binding elements enabled by the markings in \( V^{(MS,MR)}_{(4,i)} \) correspond exactly to those presented in Table 8.7 and the successor marking of each enabled binding element is also a member of \( V^{(MS,MR)}_{(4,i)} \). \( V^{(MS,MR)}_{(3a,i)} \cup V^{(MS,MR)}_{(4,i)} \cup V^{(MS,MR)}_{(1,i)} \). Thus the sublemma is proved.

These sublemmas, formulated for any \( i \in \{0, 1, ..., MS\} \), show that Tables 8.2 to 8.7 capture exactly the set of binding elements enabled by markings in \( V^{(MS,MR)}_{(4,i)} \), i.e.

\[
A^{(MS,MR)}_{(i)} = A^{(MS,MR)}_{(1,i)} \cup A^{(MS,MR)}_{(2a,i)} \cup A^{(MS,MR)}_{(2b,i)} \cup A^{(MS,MR)}_{(3a,i)} \cup A^{(MS,MR)}_{(3b,i)} \cup A^{(MS,MR)}_{(4,i)}
\]

153
8.6 Concluding Remarks

Taking the union over \( i \), \( 0 \leq i \leq MS \), gives us

\[
A_{(MS,MR)} = \bigcup_{0 \leq i \leq MS} A_i^{(MS,MR)}
\]

The sublemmas also show that for \( 0 \leq i \leq MS \), all immediate successors of all markings in \( V_i^{(MS,MR)} \) are contained in \( \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \), i.e.

\[
\forall M \in V_i^{(MS,MR)}, [M] \subseteq \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \cup V_{(1,i)}^{(MS,MR)} \cup V_{(2a,i)}^{(MS,MR)} \cup V_{(2b,i)}^{(MS,MR)} \cup V_{(3a,i)}^{(MS,MR)} \cup V_{(3b,i)}^{(MS,MR)} \cup V_{(4,i)}^{(MS,MR)} \cup V_{(1,\ell \in MS)}^{(MS,MR)} \cup V_{(4,i \in MS)}^{(MS,MR)}
\]

Taking the union over \( i \) gives

\[
\forall M \in \left( \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \right), [M] \subseteq \left( \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \right).
\]

Thus the Successor Lemma (Lemma 8.4) is proved.

For any \( MS \in \mathbb{N}^+ \) and any \( MR \in \mathbb{N} \), Lemmas 8.1 to 8.3 show that all markings in \( \bigcup_{0 \leq i \leq MS} V_i^{(MS,MR)} \) defined by Table 8.1 are reachable from the initial marking. For any \( MS \in \mathbb{N}^+ \) and any \( MR \in \mathbb{N} \), Lemma 8.4 shows that every successor of every marking in \( V_{(MS,MR)} \) is also in \( V_{(MS,MR)} \), thus proving that \( V_{(MS,MR)} \) is exactly the set of markings reachable from the initial marking. Lemma 8.4 also shows that the arcs captured by Tables 8.2 to 8.7 correspond exactly to the set of arcs with source markings in \( V_{(MS,MR)} \). Thus, for any \( MS \in \mathbb{N}^+ \) and any \( MR \in \mathbb{N} \),

\[
RG_{(MS,MR)} = (V_{(MS,MR)}, A_{(MS,MR)})
\]

and hence Theorem 8.1 is proved.

8.6 Concluding Remarks

The parametric reachability graph, \( RG_{(MS,MR)} \), is defined and proved correct in this chapter. The algebraic expressions given in Sections 8.1, 8.2 and 8.3 thus capture symbolically the concrete reachability graphs of all instantiations of the SWP CPN model. In the next chapter, we use these expressions to verify properties of the SWP for all parameter values.
Chapter 9

Analysis of the Parametric Reachability Graph

Several interesting properties of the infinite set of reachability graphs can be determined directly from the algebraic expressions in Tables 8.1 to 8.7. Consequently, these properties hold over all values of the \textit{MaxSeqNo} and \textit{MaxRetrans} parameters. Firstly, in Section 9.1, we give a proof of Conjecture 6.1 from Section 6.2 for the size of the parametric reachability graph. Then, in Sections 9.2 to 9.5, the properties for verification outlined in Section 5.7 (except for Property 5.5) are proved. Property 5.5, conformance of the protocol to its service language, is verified using language analysis in Chapter 10.

The work presented in this chapter is a revised version of that which was first presented in [50] in brief (and excluding that of Sections 9.3 and 9.4) and in [51] in full.

9.1 Size of the Reachability Graph

Conjecture 6.1 of Chapter 6 presented formulae for the size of the RG in both parameters, based on empirical evidence. We now restate this conjecture as a theorem and prove it by inspection of Tables 8.1 to 8.7.

\textbf{Theorem 9.1.} For the Stop-and-Wait CPN of Figs. 5.5 and 5.6, the number of nodes in the reachability graph is given by

\[ |V_{(MS,MR)}| = ((MS + 1)/6)(5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36) \]

and the number of arcs is given by

\[ |A_{(MS,MR)}| = ((MS + 1)/6)(30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36) \]
9.1. Size of the Reachability Graph

9.1.1 The Number of Markings

The number of nodes in the reachability graph can be derived directly from the restrictions placed on the variables for each of the 6 sets of markings represented in Table 8.1.

In the following proof we make use of formulas for the Triangular and Tetrahedral numbers [36]. For \( n \geq 1 \), the \( n^{th} \) triangular number is the sum of all integers from 1 to \( n \). The \( n^{th} \) tetrahedral number is the sum of the first \( n \) triangular numbers. Formally:

- The \( n^{th} \) triangular number is given by
  \[
  T_{rn} = 1 + 2 + 3 + \ldots + n = \sum_{k=1}^{n} k = \frac{n(n + 1)}{2} \quad (9.1)
  \]

- The \( n^{th} \) tetrahedral number is given by
  \[
  T_e n = \sum_{j=1}^{n} \left( \sum_{k=1}^{j} k \right) = \sum_{j=1}^{n} j(j + 1)/2 = \frac{n(n + 1)(n + 2)}{6} \quad (9.2)
  \]

These numbers are also given by the third and fourth diagonals of Pascal’s triangle, respectively [36].

Row 1 of Table 8.1 allows any combination of \( mo \) and \( ao \) such that \( 0 \leq mo + ao \leq MR \). Thus, for each value of \( mo \) there are \( MR + 1 - mo \) possible values of \( ao \), i.e. \( ao \in \{0, 1, \ldots, MR - mo\} \). The total number of markings of class 1 for each value of \( i \in \{0, 1, \ldots, MS\} \) is thus the sum of the number of these values for each \( mo \) between 0 and \( MR \):

\[
|V_{(1,i)}^{(MS,MR)}| = \sum_{0 \leq mo \leq MR} (MR + 1 - mo)
= (MR + 1)(MR + 1) - \sum_{0 \leq mo \leq MR} mo \quad \text{(extracting the constant term)}
= (MR + 1)^2 - (0 + 1 + 2 + 3 + \ldots + MR)
= MR^2 + 2MR + 1 - (MR(MR + 1)/2) \quad \text{(from Equation}(9.1))
= (MR^2 + 3MR + 2)/2 \quad (9.3)
\]

Row 2 of Table 8.1 (class 2a markings) specifies the same allowable combinations of \( mo \) and \( ao \) but also places restrictions on \( ret \) and \( mn \) in terms of \( ret \). Specifically, for each value of \( ret \in \{0, \ldots, MR\} \) there are \( ret + 2 \) possible values of \( mn \), i.e. \( mn \in \{0, \ldots, ret + 1\} \). Because the number of new messages is independent of the number of old messages and acknowledgements, we have \( (MR^2 + 3MR + 2)/2 \) allowable values of \( mo \) and \( ao \) in combination with the following allowable values of \( ret \) and \( mn \):

\[
\sum_{0 \leq ret \leq MR} (ret + 2) = 2(MR + 1) + \sum_{0 \leq ret \leq MR} ret \quad \text{(extracting the constant term)}
= 2(MR + 1) + MR(MR + 1)/2 \quad \text{(from Equation}(9.1))
= (MR^2 + MR)/2 + 2MR + 2
= (MR^2 + 5MR + 4)/2 \quad (9.4)
\]
Thus, the total number of class 2a markings (for each value of \(i\)) is:

\[
|V_{(2a,i)}^{(MS,MR)}| = \left(\frac{(MR^2 + 5MR + 4)}{2}\right) * \left(\frac{(MR^2 + 3MR + 2)}{2}\right) \\
= \frac{(MR^4 + 3MR^3 + 2MR^2 + 5MR^3 + 15MR^2 + 10MR + 4MR^2 + 12MR + 8)}{4} \\
= \frac{(MR^4 + 8MR^3 + 21MR^2 + 22MR + 8)}{4} \\
\]

(9.5)

Row 3 of Table 8.1 (class 2b markings) specifies that \(0 \leq ao \leq MR\) (i.e. \(MR + 1\) possible values of \(ao\)) but imposes a slightly more complicated restriction on \(mn\) and \(an\): for each \(ret \in \{0, ..., MR\}\), \(mn\) can be any number less than or equal to \(ret\) but \(mn + an \leq ret + 1\). Thus, for each value of \(ret\) there are \(ret + 1\) possible values for \(mn\). For each of these values of \(mn\) there are \(ret + 2 - mn\) possible values for \(an\). Thus, the total number of possible values of \(ret, mn\) and \(an\) is:

\[
\sum_{0 \leq ret \leq MR} \left(\sum_{0 \leq mn \leq ret} (ret + 2 - mn)\right) = \sum_{0 \leq ret \leq MR} \left((ret + 1)(ret + 2) - \sum_{0 \leq mn \leq ret} mn\right) \\
= \sum_{0 \leq ret \leq MR} ((ret + 1)(ret + 2) - ret(ret + 1)/2) \\
= \sum_{0 \leq ret \leq MR} (2(ret + 1)(ret + 2)/2 - ret(ret + 1)/2) \\
= 2(MR + 1)(MR + 2)(MR + 3)/6 - MR(MR + 1)(MR + 2)/6 \quad \text{(from Equation (9.2))} \\
= 2(MR^3 + 6MR^2 + 11MR + 6)/6 - (MR^3 + 3MR^2 + 2MR)/6 \\
= (MR^3 + 9MR^2 + 20MR + 12)/6 \\
\]

(9.6)

and the total number of class 2b markings (for each value of \(i\)) is

\[
|V_{(2b,i)}^{(MS,MR)}| = (MR + 1) * (MR^3 + 9MR^2 + 20MR + 12)/6 \\
= (MR^4 + 10MR^3 + 29MR^2 + 32MR + 12)/6 \\
\]

(9.7)

Row 4 of Table 8.1 (class 3a markings) specifies a restriction on the values of \(mo\) and \(ao\), i.e. \(0 \leq mo + ao \leq MR - 1\). This gives the following number of combinations of values of \(mo\) and \(ao\):

\[
\sum_{0 \leq mo \leq MR - 1} (MR - mo) = MR + (MR - 1) + ... + 2 + 1 \\
= MR(MR + 1)/2 = (MR^2 + MR)/2 \\
\]

(9.8)

The restrictions on values of the variables \(ret\) and \(mn\) are the same as in Row 2 (class 2a markings) and so the total number of allowable values of \(mn\) and \(ret\) is \((MR^2 + 5MR + 4)/2\) from Equation (9.4). Thus the total number of class 3a markings (for each value of \(i\)) is:

\[
|V_{(3a,i)}^{(MS,MR)}| = (MR^2 + MR)/2 * (MR^2 + 5MR + 4)/2 \\
= (MR^4 + 6MR^3 + 9MR^2 + 4MR)/4 \\
\]

(9.9)
9.1. Size of the Reachability Graph

Similarly to Row 3, Row 5 of Table 8.1 (class 3b markings) specifies that $0 \leq ao \leq MR$ (i.e. $MR + 1$ possible values of $ao$) and that for each $ret \in \{0, \ldots, MR\}$, $mn$ can be any number less than or equal to $ret$ but $mn + an \leq ret$. For each of these values of $mn$ there are $ret + 1 - mn$ possible values for $an$. This gives a total number of possible values of $ret, mn$ and $an$ of:

$$\sum_{0 \leq ret \leq MR} \left( \sum_{0 \leq mn \leq ret} (ret + 1 - mn) \right) = \sum_{0 \leq ret \leq MR} \left( (ret + 1)(ret + 1) - \sum_{0 \leq mn \leq ret} mn \right)$$

$$= \sum_{0 \leq ret \leq MR} (ret^2 + 2ret + 1 - ret(ret + 1)/2)$$

$$= \sum_{0 \leq ret \leq MR} (2ret(ret + 1)/2 + ret + 1 - ret(ret + 1)/2)$$

$$= \sum_{0 \leq ret \leq MR} (ret(ret + 1)/2 + ret + 1)$$

$$= MR(MR + 1)(MR + 2)/6 + (MR + 1)(MR + 2)/2$$

$$= (MR^2 + 3MR^2 + 2MR)/6 + (3MR^2 + 9MR + 6)/6$$

$$= (MR^3 + 6MR^2 + 11MR + 6)/6$$

Thus, giving the total number of class 3b markings (for each value of $i$) of:

$$|V_{(3b,i)}^{(MS,MR)}| = (MR + 1) * (MR^2 + 6MR^2 + 11MR + 6)/6$$

$$= (MR^3 + 7MR^3 + 17MR^2 + 17MR + 6)/6$$

(9.11)

The last row of Table 8.1 (class 4 markings) specifies that $0 \leq mo + ao \leq MR - 1$, i.e. there are $MR$ possible values of $mo$. For each $mo \in \{0, \ldots, MR - 1\}$ there are $MR - mo$ possible values of $ao$, i.e. $ao \in \{0, \ldots, MR - 1 - mo\}$. The total number of markings of class 4 (for each value of $i$) is thus:

$$|V_{(4,i)}^{(MS,MR)}| = \sum_{0 \leq mo \leq MR - 1} (MR - mo) = MR + (MR - 1) + \ldots + 1$$

$$= MR(MR + 1)/2 = (MR^2 + MR)/2$$

(9.12)

The total number of markings in $V_{i}^{(MS,MR)}$, $0 \leq i \leq MS$ is the summation of Equations (9.3), (9.5), (9.7), (9.9), (9.11) and (9.12), i.e.

$$|V_{i}^{(MS,MR)}| = |V_{(1,i)}^{(MS,MR)}| + |V_{(2a,i)}^{(MS,MR)}| + |V_{(2b,i)}^{(MS,MR)}| + |V_{(3a,i)}^{(MS,MR)}| + |V_{(3b,i)}^{(MS,MR)}| + |V_{(4,i)}^{(MS,MR)}|$$

$$= (MR^2 + 3MR^2 + 2MR)/2 + (MR^4 + 8MR^3 + 21MR^2 + 22MR + 8)/4$$

$$+ (MR^4 + 10MR^3 + 29MR^2 + 32MR + 12)/6$$

$$+ (MR^4 + 6MR^3 + 9MR^2 + 4MR)/4$$

$$+ (MR^4 + 7MR^3 + 17MR^2 + 17MR + 6)/6 + (MR^2 + MR)/2$$

158
9.1. Size of the Reachability Graph

\[
9.1. \text{Size of the Reachability Graph} = \frac{6MR^2 + 12MR + 6}{6} + \frac{3MR^4 + 21MR^3 + 45MR^2 + 39MR + 12}{6} \\
+ \frac{2MR^4 + 17MR^3 + 46MR^2 + 49MR + 18}{6} = \frac{5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36}{6}
\]

Given that there are the same number of nodes in \(V_{i}^{(MS, MR)}\) for each value of \(i \in \{0, 1, ..., MS\}\), we multiply the above by \((MS + 1)\) to get the total number of nodes in \(V_{(MS, MR)}\):

\[
|V_{(MS, MR)}| = (MS + 1)(5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36)/6 \quad (9.13)
\]

9.1.2 The Number of Arcs

The number of arcs in the reachability graph can be derived directly from the restrictions placed on the variables for each of the sets of arcs in Tables 8.2 to 8.7. To formulate the total number of arcs, we first identify the number of markings which do not satisfy the given conditions for each row in the corresponding table, and then subtract these from \(|V_{(class, i)}^{(MS, MR)}|\) for each corresponding \(class \in \{1, 2a, 2b, 3a, 3b, 4\}\) to obtain the number of arcs defined by each row. Finally, the number of arcs defined by each row are added together. This method is used to determine the number of arcs defined by each of Tables 8.2 to 8.7.

The reason for using this method is that the number of markings that do not satisfy the conditions for each row of each arc table is smaller than the number of markings that do, and is thus easier to formulate.

**Outgoing Arcs from Class 1 Markings**

Table 8.2 defines the outgoing arcs of all nodes in \(V_{(1, i)}^{(MS, MR)}\) for each \(i \in \{0, ..., MS\}\), where \(0 \leq mo + ao \leq MR\). The New Restrictions column gives details of the restrictions on the five variables \(mo, ao, mn, an\) and \(ret\) in addition to those specified by the set \(V_{(1, i)}^{(MS, MR)}\) itself. Row 1 enforces no extra restrictions, thus row 1 defines one outgoing arc for every marking in \(V_{(1, i)}^{(MS, MR)}\). Rows 2 and 3 specify that \(mo \geq 1\), i.e. \(1 \leq mo + ao \leq MR, mo \geq 1\). Rows 4 and 5 specify that \(ao \geq 1\), i.e. \(1 \leq mo + ao \leq MR, ao \geq 1\).

The number of arcs defined by row 1 for each \(i \in \{0, ..., MS\}\) is thus \(|V_{(1, i)}^{(MS, MR)}| = (MR^2 + 3MR + 2)/2\), from Equation (9.3).

When \(mo = 0\) there are \(MR + 1\) possible values of \(ao\), i.e. \(ao \in \{0, ..., MR\}\). This gives \(MR + 1\) markings in \(V_{(1, i)}^{(MS, MR)}\) that satisfy \(mo = 0\) and hence don’t satisfy \(mo > 0\). The number of arcs defined by each of rows 2 and 3 becomes \(|V_{(1, i)}^{(MS, MR)}| - (MR + 1)\), for each \(i \in \{0, ..., MS\}\).

Similarly, when \(ao = 0\) there are \(MR + 1\) possible values of \(mo\). Thus the number of arcs defined by each of rows 4 and 5 is also \(|V_{(1, i)}^{(MS, MR)}| - (MR + 1)\).
The total number of arcs defined by Table 8.2, when substituting \( |V^{(MS,MR)}_{(1,i)}| \) from Equation 9.3 becomes:

\[
|A^{(MS,MR)}_{(1,i)}| = 5|V^{(MS,MR)}_{(1,i)}| - 4(MR + 1)
\]

\[
= 5(MR^2 + 3MR + 2)/2 - 4(MR + 1)
\]

\[
= (5MR^2 + 7MR + 2)/2
\]

(9.14)

**Outgoing Arcs from Class 2a Markings**

Table 8.3 defines all outgoing arcs of the nodes in \( V^{(MS,MR)}_{(2a,i)} \) for each \( i \in \{0, ..., MS\} \), where \( 0 \leq mo + ao \leq MR, 0 \leq ret \leq MR \) and \( 0 \leq mn \leq ret + 1 \). From the New Restrictions column, only nodes in \( V^{(MS,MR)}_{(2a,i)} \) that satisfy \( ret = MR \) do not have arcs defined by row 1. Rows 2 and 4 define arcs only when \( mo \geq 1 \). Row 3 defines arcs when \( mn \geq 1 \) and row 5 defines arcs when \( mo = 0 \) and \( mn \geq 1 \). Row 6 and 7 define arcs only when \( ao \geq 1 \).

When \( ret = MR \) there are \((MR^2 + 3MR + 2)/2\) allowable combinations of \( mo \) and \( ao \) (from Equation (9.3)) and \( MR + 2 \) allowable values of \( mn \), thus giving \((MR^3 + 5MR^2 + 8MR + 4)/2\) nodes in \( V^{(MS,MR)}_{(2a,i)} \) in which \( ret = MR \). Hence, \(|V^{(MS,MR)}_{(2a,i)}| - (MR^3 + 5MR^2 + 8MR + 4)/2\) nodes in \( V^{(MS,MR)}_{(2a,i)} \) have outgoing arcs defined by row 1.

When \( mo = 0 \) there are \( MR + 1 \) possible values of \( ao \) and \((MR^2 + 5MR + 4)/2\) possible combinations of values of \( ret \) and \( mn \) (from Equation (9.4)), giving \((MR^3 + 6MR^2 + 9MR + 4)/2\) such nodes. The number of nodes with outgoing arcs defined by each of rows 2 and 4 is thus \(|V^{(MS,MR)}_{(2a,i)}| - (MR^3 + 6MR^2 + 9MR + 4)/2\).

When \( mn = 0 \) there are \((MR^2 + 3MR + 2)/2\) allowable combinations of \( mo \) and \( ao \) (Equation (9.3)) and \( MR + 1 \) possible values of \( ret \), giving \((MR^3 + 4MR^2 + 5MR + 2)/2\) markings in \( V^{(MS,MR)}_{(2a,i)} \) with \( mn = 0 \). Hence, there are \(|V^{(MS,MR)}_{(2a,i)}| - (MR^3 + 4MR^2 + 5MR + 2)/2\) arcs defined by row 3.

The conditions under which a node in \( V^{(MS,MR)}_{(2a,i)} \) will not have an outgoing arc defined by row 5 is when \( mo \geq 1 \) or \( mn = 0 \). Note that this is a logical or, not and, of conditions. This corresponds to the following two non-overlapping situations: \( mo \geq 1 \) for \( mn \geq 0 \), and \( mo \geq 0 \) for \( mn = 0 \). Here, the conditions under which no arc is defined must be expressed as non-overlapping situations to ensure markings are counted exactly once.

When \( mo = 0 \) there are \( MR + 1 \) possible values of \( ao \). When \( mn = 0 \), there are \( MR + 1 \) possible values of \( ret \). This gives \((MR + 1)(MR + 1)\) markings that satisfy \( mo = 0 \) and \( mn = 0 \). When \( mo \geq 1, 1 \leq mo + ao \leq MR \), there are \((MR^2 + MR)/2\) possible allowable combinations of \( mo \) and \( ao \), using Equation (9.12) as done previously. When there are no restrictions on \( mn \) there are \( ret + 2 \) possible values of \( mn \) for each \( ret \in \{0, ..., MR\} \), giving \((MR^2 + 5MR + 4)/2\) allowable combinations of \( mn \) and \( ret \) (from Equation (9.4)). This gives \((MR^2 + 5MR + 4)(MR^2 + MR)/4\) markings that satisfy \( mo \geq 1 \).
Hence, there are \((MR + 1)(MR + 1) + (MR^2 + 5MR + 4)(MR^2 + MR)/4\) markings in \(V_{(2a,i)}^{(MS,MR)}\) that satisfy \(mo \geq 1\) or \(mn = 0\). This gives \(|V_{(2a,i)}^{(MS,MR)}| - (MR^4 + 6MR^3 + 13MR^2 + 12MR + 4)/4\) arcs defined by row 5.

When \(ao = 0\) there are \(MR + 1\) possible values of \(mo\) and \((MR^2 + 5MR + 4)/2\) possible combinations of values of \(ret\) and \(mn\) (from Equation (9.4)), giving \((MR^3 + 6MR^2 + 9MR + 4)/2\) such nodes. The number of nodes with outgoing arcs defined by each of rows 6 and 7 is thus \(|V_{(2a,i)}^{(MS,MR)}| - (MR^3 + 6MR^2 + 9MR + 4)/2\).

Adding up the number of arcs defined by rows 1 to 7 and substituting \(|V_{(2a,i)}^{(MS,MR)}|\) from Equation (9.5) gives:

\[
|A_{(2a,i)}^{(MS,MR)}| = 7|V_{(2a,i)}^{(MS,MR)}| - (MR^3 + 5MR^2 + 8MR + 4)/2
- 4(MR^3 + 6MR^2 + 9MR + 4)/2 - (MR^3 + 4MR^2 + 5MR + 2)/2
- (MR^4 + 6MR^3 + 13MR^2 + 12MR + 4)/4
= 7(MR^4 + 8MR^3 + 21MR^2 + 22MR + 8)/4
- (MR^4 + 18MR^3 + 79MR^2 + 110MR + 48)/4
= (6MR^4 + 38MR^3 + 68MR^2 + 44MR + 8)/4
\]  

\((9.15)\)

**Outgoing Arcs from Class 2b Markings**

Table 8.4 defines all outgoing arcs of the nodes in \(V_{(2b,i)}^{(MS,MR)}\), where \(0 \leq ao \leq MR, 0 \leq ret \leq MR, 0 \leq mn \leq ret\) and \(0 \leq mn + an \leq ret + 1\). Looking at the New Restrictions column, row 1 defines outgoing arcs for markings satisfying \(ret < MR\), rows 2 and 3 for \(mn \geq 1\), rows 4 and 6 for \(ao \geq 1\), row 5 for \(an \geq 1\) and row 7 for \(ao = 0\) and \(an \geq 1\).

When \(ret = MR\) there are \(MR + 1\) possible values of \(mn\), i.e. \(mn \in \{0, ..., MR\}\), each of which allows \(MR + 2 - mn\) possible values of \(an\), i.e. \(an \in \{0, ..., MR + 1 - mn\}\). This gives a total number of combinations of \(mn\) and \(an\) of \((MR + 1)(MR + 2) - MR(MR + 1)/2 = (MR^2 + 5MR + 4)/2\), from the inner sum of Equation (9.6) by replacing \(ret\) with \(MR\). Combined with the \(MR + 1\) possible values of \(ao\) this gives \(|V_{(2b,i)}^{(MS,MR)}| - (MR^3 + 6MR^2 + 9MR + 4)/2\) arcs defined in row 1.

There are \(MR + 1\) possible values of \(ret\) and when \(mn = 0\) there are \(ret + 2\) possible values of \(an\) for each value of \(ret \in \{0, ..., MR\}\). This gives \((MR^2 + 5MR + 4)/2\) possible values of \(ret\) and \(an\) (from Equation (9.4) when considering \(an\) and not \(mn\)). When combined with the \(MR + 1\) possible values of \(ao\), this gives \(|V_{(2b,i)}^{(MS,MR)}| - (MR^3 + 6MR^2 + 9MR + 4)/2\) arcs defined by each of rows 2 and 3.

Rows 4 and 6 define arcs only when \(ao \geq 1\), i.e. not when \(ao = 0\). From Equation (9.6) there are \((MR^3 + 9MR^2 + 20MR + 12)/6\) allowable combinations of \(ret, mn\) and \(an\), and so the number of
arcs defined by each of rows 4 and 6 is $|V_{(2b,i)}^{(MS,MR)}| - (MR^3 + 9MR^2 + 20MR + 12)/6$.

When $an = 0$ there are $ret + 1$ possible values of $mn$ for each value of $ret \in \{0,...,MR\}$. (The expression $0 \leq mn + an \leq ret + 1$ becomes superfluous when $an = 0$.) The total number of combinations of values of $ret$ and $mn$ is thus

$$\sum_{0 \leq ret \leq MR} ret + 1 = (MR + 1)(MR + 2)/2$$  \hspace{1cm} (9.16)

and when combined with the $MR + 1$ possible values of $ao$, gives $(MR + 1)(MR + 2)/2$ nodes in $V_{(2b,i)}^{(MS,MR)}$ for which row 5 does not define an outgoing arc. Hence the number of arcs defined by row 5 is $|V_{(2b,i)}^{(MS,MR)}| - (MR^3 + 4MR^2 + 5MR + 2)/2$.

Row 7 defines arcs only for $ao = 0$ and $an \geq 1$, i.e. not for nodes in which $ao \geq 1$ or $an = 0$. Again, care must be taken with the logical or. Nodes without an outgoing arc defined by row 7 correspond to one of the following three situations: $ao \geq 1$ and $an \geq 1$, $ao = 0$ and $an = 0$, or $ao \geq 1$ and $an = 0$. Combining the first and third cases gives the simplified restrictions of $ao = 0$ and $an = 0$, and $ao \geq 1$ for any $an$.

When $ao = 0$ there is only one value of $mo$ as $mo = 0$ by definition. When $an = 0$ there are $ret + 1$ possible values of $mn$ for each value of $ret \in \{0,...,MR\}$. From Equation (9.16), this gives $(MR + 1)(MR + 2)/2$ possible combinations of values of $mn$ and $ret$, and also $(MR + 1)(MR + 2)/2$ nodes in which $ao = 0$ and $an = 0$. When $ao \geq 1$ there are $MR$ possible values of $ao$. When there are no restrictions on $an$ there are $(MR^3 + 9MR^2 + 20MR + 12)/6$ possible combinations of $mn, an$ and $ret$ (from Equation (9.6)). This gives $MR(MR^3 + 9MR^2 + 20MR + 12)/6$ nodes that satisfy $ao \geq 1$. Thus, row 7 defines $|V_{(2b,i)}^{(MS,MR)}| - (MR^4 + 9MR^3 + 20MR^2 + 12MR)/6 - (MR^2 + 3MR + 2)/2$ arcs.

Adding up the number of arcs defined by rows 1 to 7 and substituting $|V_{(2b,i)}^{(MS,MR)}|$ from Equation (9.7) gives:

$$|A_{(2b,i)}^{(MS,MR)}| = 7|V_{(2b,i)}^{(MS,MR)}| - 3(MR^3 + 6MR^2 + 9MR + 4)/2$$

$$- 2(MR^3 + 9MR^2 + 20MR + 12)/6 - (MR^3 + 4MR^2 + 5MR + 2)/2$$

$$- (MR^4 + 9MR^3 + 20MR^2 + 12MR)/6 - (MR^2 + 3MR + 2)/2$$

$$= 7(MR^4 + 10MR^3 + 29MR^2 + 32MR + 12)/6$$

$$- (4MR^3 + 23MR^2 + 35MR + 16)/2$$

$$- (MR^4 + 11MR^3 + 38MR^2 + 52MR + 24)/6$$

$$= (6MR^4 + 47MR^3 + 96MR^2 + 67MR + 12)/6$$
9.1. Size of the Reachability Graph

Outgoing Arcs from Class 3a Markings

Table 8.5 defines all outgoing arcs of the nodes in $V_{(3a,i)}^{(MS,MR)}$ for each $i \in \{0, ..., MS\}$, where $0 \leq mo + ao \leq MR - 1$, $0 \leq ret \leq MR$ and $0 \leq mn \leq ret + 1$. The New Restrictions column specifies that row 1 defines arcs only for nodes satisfying $ret < MR$, row 2 for $mo \geq 1$, row 3 for $mn \geq 1$ and rows 4 and 5 for $ao \geq 1$. Row 6 defines an arc for every marking in $V_{(3a,i)}^{(MS,MR)}$.

When $ret = MR$, there are $MR + 2$ possible values of $mn$, i.e. $mn \in \{0, ..., MR + 1\}$. From Equation (9.8) there are $(MR^2 + MR)/2$ possible combinations of $mo$ and $ao$. Hence row 1 defines $|V_{(3a,i)}^{(MS,MR)}| - (MR^3 + 3MR^2 + 2MR)/2$ arcs.

With the restriction of $mo = 0$ there are $MR$ possible values of $ao$, i.e. $ao \in \{0, ..., MR - 1\}$. From Equation (9.4) there are $(MR^2 + 5MR + 4)/2$ allowable combinations of $mn$ and $ret$. Thus there are $MR(MR^2 + 5MR + 4)/2$ nodes in $V_{(3a,i)}^{(MS,MR)}$ for which row 2 does not define an arc, and thus row 2 defines $|V_{(3a,i)}^{(MS,MR)}| - (MR^3 + 5MR^2 + 4MR)/2$ arcs.

With the restriction $mn = 0$ there are $MR + 1$ possible values of $ret$. Combined with the $(MR^2 + MR)/2$ possible values of $mo$ and $ao$ (from Equation (9.8)) row 3 defines $|V_{(3a,i)}^{(MS,MR)}| - (MR^3 + 2MR^2 + MR)/2$ arcs.

With the restriction $ao = 0$ there are $MR$ possible values of $mo$. Given that there are $(MR^2 + 5MR + 4)/2$ allowable combinations of $ret$ and $mn$ (from Equation (9.4)) rows 4 and 5 each define $|V_{(3a,i)}^{(MS,MR)}| - (MR^3 + 5MR^2 + 4MR)/2$ arcs.

The total number of arcs defined by Table 8.5 is obtained by summing the totals for each row and substituting for $|V_{(3a,i)}^{(MS,MR)}|$ from Equation (9.9):

$$|A_{(3a,i)}^{(MS,MR)}| = 6|V_{(3a,i)}^{(MS,MR)}| - (MR^3 + 3MR^2 + 2MR)/2 - (MR^3 + 5MR^2 + 4MR)/2$$
$$- (MR^3 + 2MR^2 + MR)/2 - 2(MR^3 + 5MR^2 + 4MR)/2$$
$$= 6(MR^4 + 6MR^3 + 9MR^2 + 4MR)/4 - (10MR^3 + 40MR^2 + 30MR)/4$$
$$= (6MR^4 + 26MR^3 + 14MR^2 - 6MR)/4$$

(9.18)

Outgoing Arcs from Class 3b Markings

Table 8.6 defines all outgoing arcs of the nodes in $V_{(3b,i)}^{(MS,MR)}$ for each $i \in \{0, ..., MS\}$, where $0 \leq ao \leq MR$, $0 \leq ret \leq MR$ and $0 \leq mn + an \leq ret$. Row 1 defines an outgoing arc only if $ret < MR$, row 2 only if $mn \geq 1$, rows 3 and 5 only if $ao \geq 1$, row 4 only if $an \geq 1$, and row 6 only if $ao = 0$ and $an \geq 1$. Row 7 defines an outgoing arc for every node in $V_{(3b,i)}^{(MS,MR)}$.

When $ret = MR$ there are $(MR^2 + 3MR + 2)/2$ allowable combinations of $mn$ and $an$, obtained by replacing $mo$ and $ao$ with $mn$ and $an$ in Equation (9.3). Combined with the $MR + 1$ allowable values of $ao$, row 1 defines $|V_{(3b,i)}^{(MS,MR)}| - (MR^3 + 4MR^2 + 5MR + 2)/2$ arcs.
9.1. Size of the Reachability Graph

When \( mn = 0 \) there are \( ret + 1 \) possible values of \( an \) for each \( ret \in \{0, \ldots, MR\} \). This gives \( \sum_{0 \leq ret \leq MR} (ret + 1) = (MR + 1)(MR + 2)/2 \) allowable combinations of \( an \) and \( ret \) from Equation (9.16). Combined with the \( MR + 1 \) possible values of \( ao \), row 2 defines \( |V_{(3b,i)}^{(MS,MR)}| - (MR^3 + 4MR^2 + 5MR + 2)/2 \) arcs.

The value of \( ao \) does not influence the possible values of \( mn, an \) or \( ret \), so if \( ao = 0 \) there are still \( (MR^3 + 6MR^2 + 11MR + 6)/6 \) allowable combinations of \( mn, an \) and \( ret \) (from Equation (9.10)). Therefore rows 3 and 5 each define \( |V_{(3b,i)}^{(MS,MR)}| - (MR^3 + 6MR^2 + 11MR + 6)/6 \) arcs.

Row 4 does not define arcs when \( an = 0 \). This gives \( ret + 1 \) possible values of \( mn \) for each value of \( ret \in \{0, \ldots, MR\} \), i.e. \( \sum_{0 \leq ret \leq MR} (ret + 1) = (MR + 1)(MR + 2)/2 \) allowable combinations of \( mn \) and \( ret \). Combined with the \( MR + 1 \) allowable values of \( ao \), row 4 defines \( |V_{(3b,i)}^{(MS,MR)}| - (MR^3 + 4MR^2 + 5MR + 2)/2 \) arcs.

Row 6 will not define arcs if \( ao \geq 1 \) or \( an = 0 \). This corresponds to the two non-overlapping situations of \( ao = 0 \) and \( an = 0 \), and \( ao \geq 1 \) for all allowable values of \( an \).

In the first situation, when \( an = 0 \) there are \( ret + 1 \) allowable values of \( mn \) for each \( ret \in \{0, \ldots, MR\} \), i.e. \( (MR + 1)(MR + 2)/2 \) allowable combinations of \( mn \) and \( ret \) as above. Thus for the single allowable value of \( ao = 0 \) there are \( (MR^2 + 3MR + 2)/2 \) nodes satisfying the condition \( ao = 0 \) and \( an = 0 \).

In the second situation, for \( ao \geq 1 \), there are \( MR \) allowable values of \( ao \) and \( (MR^3 + 6MR^2 + 11MR + 6)/6 \) allowable combinations of \( mn, an \) and \( ret \) (from Equation (9.10)). This gives a total of \( (MR^4 + 6MR^3 + 11MR^2 + 6MR)/6 \) nodes in \( V_{(3b,i)}^{(MS,MR)} \) that satisfy the condition of \( ao \geq 1 \).

The total number of arcs defined by row 6 is thus \( |V_{(3b,i)}^{(MS,MR)}| - (MR^4 + 6MR^3 + 11MR^2 + 6MR)/6 - (3MR^2 + 9MR + 6)/6 = |V_{(3b,i)}^{(MS,MR)}| - (MR^4 + 6MR^3 + 14MR^2 + 15MR + 6)/6 \) arcs.

The total number of arcs defined by Table 8.6, when substituting for \( |V_{(3b,i)}^{(MS,MR)}| \) from Equation (9.11) is given by:

\[
\]
Outgoing Arcs from Class 4 Markings

Table 8.7 defines all outgoing arcs of the nodes in $V^{(MS,MR)}_{(4,i)}$ for each $i \in \{0, ..., MS\}$, where $0 \leq mo + ao \leq MR - 1$. From the New Restrictions column, row 2 defines arcs only for nodes with $mo \geq 1$ and rows 3 and 4 define arcs only for nodes with $ao \geq 1$. Rows 1 and 5 each define an arc for every marking in $V^{(MS,MR)}_{(4,i)}$.

When $mo = 0$ there are $MR$ allowable values of $ao$, thus row 2 defines $|V^{(MS,MR)}_{(4,i)}| - MR$ arcs. When $ao = 0$ there are $MR$ allowable values of $mo$, thus rows 3 and 4 each define $|V^{(MS,MR)}_{(4,i)}| - MR$ arcs. The total number of arcs defined by Table 8.7 is given by:

$$|A^{(MS,MR)}_{(4,i)}| = 5|V^{(MS,MR)}_{(4,i)}| - 3MR$$

Substituting for $|V^{(MS,MR)}_{(4,i)}|$ using Equation (9.12) we obtain

$$= 5(MR^2 + MR)/2 - 3MR$$

$$(9.20)$$

The Total Number of Arcs

The total number of arcs in $A^{(MS,MR)}_{i}$ is thus the summation of Equations (9.14), (9.15), (9.17), (9.18), (9.19), and (9.20):

$$|A^{(MS,MR)}_{i}| = |A^{(MS,MR)}_{(1,i)}| + |A^{(MS,MR)}_{(2a,i)}| + |A^{(MS,MR)}_{(2b,i)}|$$

$$+ |A^{(MS,MR)}_{(3a,i)}| + |A^{(MS,MR)}_{(3b,i)}| + |A^{(MS,MR)}_{(4,i)}|$$

$$= (5MR^2 + 7MR + 2)/2 + (6MR^4 + 38MR^3 + 68MR^2 + 44MR + 8)/4$$

$$+ (6MR^4 + 47MR^3 + 96MR^2 + 67MR + 12)/6$$

$$+ (6MR^4 + 26MR^3 + 14MR^2 - 6MR)/4$$

$$+ (6MR^4 + 32MR^3 + 57MR^2 + 37MR + 6)/6 + (5MR^2 - MR)/2$$

$$= (10MR^2 + 6MR + 2)/2 + (12MR^4 + 64MR^3 + 82MR^2 + 38MR + 8)/4$$

$$+ (12MR^4 + 79MR^3 + 153MR^2 + 104MR + 18)/6$$

$$= (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)/6$$

As with nodes, there are the same number of arcs in $A^{(MS,MR)}_{i}$ for each $i \in \{0, 1, ..., MS\}$. Hence, we multiply the above by $(MS + 1)$ to get the total number of arcs in $A^{(MS,MR)}$:

$$|A^{(MS,MR)}| = \sum_{0 \leq i \leq MS} ((30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)/6)$$

$$= (MS + 1)(30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)/6$$

By Equations (9.13) and (9.21) Theorem 9.1 is proved.
9.2 Absence of Unexpected Dead Markings

This is Property 5.1 of Section 5.7. Each table of arcs captures the arcs with a source marking in one of the 6 sets of nodes defined in Table 8.1. For each table of arcs, if a reachable marking exists in the corresponding set of markings that does not satisfy at least one of the restrictions in the New Restrictions column, then this is a dead marking. For example, consider $M_{(2a,0),(0,0,0,0,MR)}^{(MS,MR)} \in V_{(2a,0)}^{(MS,MR)}$. This does not satisfy any of the restrictions in the New Restrictions column of Table 8.3 ($ret \not< MR$, $mo \not< 1$, $mn \not< 1$, and $ao \not< 1$). Hence, this is a dead marking.

Once the dead markings are known, they must be examined to determine which of these are expected, and which are not. All nodes in $V_{(1,i)}^{(MS,MR)}$, $V_{(3a,i)}^{(MS,MR)}$, $V_{(3b,i)}^{(MS,MR)}$ and $V_{(4,i)}^{(MS,MR)}$ have at least one outgoing arc, indicated by there being no restrictions in at least one row of their arc tables (i.e. row 1 of Table 8.2, row 6 of Table 8.5, row 7 of Table 8.6 and rows 1 and 5 of Table 8.7 respectively), for any $0 \leq i \leq MS$, $MS \in \mathbb{N}^+$. This leaves the possibility that dead markings exist in the sets $V_{(2a,i)}^{(MS,MR)}$ and $V_{(2b,i)}^{(MS,MR)}$, for $0 \leq i \leq MS$, $MS \in \mathbb{N}^+$.

Examination of the New Restrictions column in Table 8.3 reveals that any marking that does not satisfy any of $ret < MR$, $mo > 0$, $mn > 0$ or $ao > 0$ is a dead marking. This corresponds to the subset of $V_{(2a,i)}^{(MS,MR)}$ such that $ret = MR$, $mo = 0$, $mn = 0$ and $ao = 0$ (recalling that $an = 0$ by definition of $V_{(2a,i)}^{(MS,MR)}$), i.e. a marking in which the sender is waiting for an acknowledgement of a message with sequence number $i$, the receiver is in the ready state and expecting a message with sequence number $i$, both the message and acknowledgement channels are empty, and the retransmission counter has reached its limit. This corresponds to the loss of the original message plus all retransmissions of the message with sequence number $i$. This is just one marking for each value of the sender sequence number, which we define as the set

$$V_{(dead,2a)}^{(MS,MR)} = \{ M_{(2a,i),(0,0,0,0,MR)}^{(MS,MR)} \mid 0 \leq i \leq MS \} \quad (9.22)$$

A similar examination of the Restrictions column in Table 8.4 reveals that any marking that does not satisfy $ret < MR$, $mn > 0$, $ao > 0$ or $an > 0$ is a dead marking. This corresponds to the subset of $V_{(2b,i)}^{(MS,MR)}$ such that $ret = MR$, $mn = 0$, $ao = 0$ and $an = 0$ (recalling that $mo = 0$ by definition of $V_{(2b,i)}^{(MS,MR)}$), i.e. a marking in which the sender is waiting for an acknowledgement of a message with sequence number $i$, the receiver is in the ready state and expecting a message with sequence number $i \oplus MS 1$, both the message and acknowledgement channels are empty, and the retransmission counter has reached its limit. This corresponds to the reception of at least one of the messages with sequence number $i$ but a combination of loss of other messages with sequence number $i$ and loss of all acknowledgements of this message. Again, this is just one marking per sender sequence number, which we define as the set

$$V_{(dead,2b)}^{(MS,MR)} = \{ M_{(2b,i),(0,0,0,0,MR)}^{(MS,MR)} \mid 0 \leq i \leq MS \} \quad (9.23)$$
The complete set of dead markings for any combination of parameters \(MS \in \mathbb{N}^+\) and \(MR \in \mathbb{N}\) is thus:

\[
V_{\text{dead}}^{(MS,MR)} = \{M^{(MS,MR)}_{(2a,i),(0,0,0,0,MR)}, M^{(MS,MR)}_{(2b,i),(0,0,0,0,MR)} | 0 \leq i \leq MS\} \quad (9.24)
\]

From Table 7.2 each class 2a marking satisfies the properties of \(M(\text{sender\_state}) = 1\_\text{waiting}, M(\text{receiver\_state}) = 1\_\text{ready}\) and \(M(\text{send\_seq\_no}) = M(\text{recv\_seq\_no})\). Thus, by inspection, each class 2a marking in \(V_{\text{dead}}^{(MS,MR)}\) is an element of \(V_{\text{term1}}\) as defined in Property 5.1 in Section 5.7. Similarly, each class 2b marking in \(V_{\text{dead}}^{(MS,MR)}\) is an element of \(V_{\text{term2}}\), also given in Property 5.1 in Section 5.7. Hence there are no unexpected dead markings and Property 5.1 is verified.

### 9.3 Absence of Livelock

This is Property 5.2 of Section 5.7. As discussed in Section 5.7 a sufficient condition for absence of livelock (without the need to reason about strongly connected components) is that all markings can reach a dead marking.

**Lemma 9.1.** All markings in the parametric reachability graph of the Stop-and-Wait CPN of Figs. 5.5 and 5.6 can reach a dead marking, i.e. \(\forall M^{(MS,MR)}_{(\text{class},i),(\text{mo,ao,mn,an,ret})} \in V_{(MS,MR)} \exists M \in V_{\text{dead}}^{(MS,MR)}\) such that \(M \in \{M^{(MS,MR)}_{(\text{class},i),(\text{mo,ao,mn,an,ret})}\}, \text{where class} \in \{1, 2a, 2b, 3a, 3b, 4\}\) and \(V_{\text{dead}}^{(MS,MR)}\) is given by Equation (9.24).

**Proof.** This lemma is proved in two parts. Firstly, we must show that all markings in each \(V_{(2a,i)}^{(MS,MR)}\)

and \(V_{(2b,i)}^{(MS,MR)}\) can reach the corresponding dead marking, \(M^{(MS,MR)}_{(2a,i),(0,0,0,0,MR)}\) and \(M^{(MS,MR)}_{(2b,i),(0,0,0,0,MR)}\) respectively, for each value of \(i, 0 \leq i \leq MS\). Consider an arbitrary marking, \(M^{(MS,MR)}_{(\text{class},i),(\text{mo,ao,mn,an,ret})} \in V_{(2a,i)}^{(MS,MR)}\). By Lemma 6.3, the \text{timeout\_retrans} transition can occur repeatedly until the maximum number of retransmissions is reached, i.e. \(MR - \text{ret}\) times. This leads to a marking \(M^{(MS,MR)}_{(\text{class},i),(\text{mo,ao,mn+MR\_ret,0,MR})}\). From Lemmas 6.8 and 6.9, the \text{mess\_loss} and \text{ack\_loss} transitions can occur repeatedly and in any order until both the message and acknowledgement channel are empty, resulting in the dead marking \(M^{(MS,MR)}_{(\text{class},i),(0,0,0,0,MR)}\). This holds for all markings in \(V_{(2a,i)}^{(MS,MR)}\). The argument is identical for demonstrating the ability of all markings \(M^{(MS,MR)}_{(\text{class},i),(\text{mo,ao,mn,an,ret})} \in V_{(2b,i)}^{(MS,MR)}\) to be able to reach the dead marking \(M^{(MS,MR)}_{(\text{class},i),(0,0,0,0,MR)}\). By the sequence number independent behaviour captured in Lemmas 6.2 to 6.9, this result is valid for all \(i, 0 \leq i \leq MS\). Therefore, all class 2a and 2b markings can reach a dead marking.

Secondly, we must show that all other markings (of class 1, 3a, 3b and 4) can reach either a class 2a or class 2b marking:

**Class 1 markings:** From row 1 of Table 8.2, all markings in \(V_{(1,i)}^{(MS,MR)}\) enable the \text{send\_mess} transition, thus all markings in \(V_{(1,i)}^{(MS,MR)}\) can reach a marking in \(V_{(2a,i)}^{(MS,MR)}\).
9.4. Absence of Unexpected Dead Transitions

**Class 3a markings:** From row 6 of Table 8.5, all markings in \( V^{(MS,MR)}_{(3a,i)} \) enable the `send_ack` transition, thus all markings in \( V^{(MS,MR)}_{(3a,i)} \) can reach a marking in \( V^{(MS,MR)}_{(2a,i)} \).

**Class 3b markings:** From row 7 of Table 8.6, all markings in \( V^{(MS,MR)}_{(3b,i)} \) enable the `send_ack` transition, thus all markings in \( V^{(MS,MR)}_{(3b,i)} \) can reach a marking in \( V^{(MS,MR)}_{(2b,i)} \).

**Class 4 markings:** Finally, from row 5 of Table 8.7, all markings in \( V^{(MS,MR)}_{(4,i)} \) enable the `send_ack` transition, thus all markings in \( V^{(MS,MR)}_{(4,i)} \) can reach a marking in \( V^{(MS,MR)}_{(1,i)} \), and we know already that from row 1 of Table 8.2, all markings in \( V^{(MS,MR)}_{(1,i)} \) can reach a marking in \( V^{(MS,MR)}_{(2a,i)} \).

Again, Lemmas 6.2 to 6.9 mean that this result is valid for all values of \( i, 0 \leq i \leq MS \). Thus all markings in \( V^{(MS,MR)} \) can reach a dead marking and the lemma is proved.

That all markings can reach a dead marking implies there are no livelocks and hence Property 5.2 is verified.

### 9.4 Absence of Unexpected Dead Transitions

This is Property 5.3 of Section 5.7. This property states that every transition is enabled by at least one reachable marking, except for the `timeout_retrans` and `receive_dup_ack` transitions when \( MR = 0 \).

This is proved by inspection of the arc tables, Tables 8.2 to 8.7. Each transition appears in at least one row of at least one of the Tables 8.2 to 8.7 and is enabled by at least one marking, regardless of the values of the parameters. Giving one such example for each transition, and by direct inspection of the CPN model in Fig. 5.5:

**send_mess** is enabled by the initial marking, \( M^{(MS,MR)}_{(1,0),(0,0,0,0,0)} \), which is present in every concrete instance of the parametric reachability graph.

**receive_mess** and **mess_loss** are enabled by the marking \( M^{(MS,MR)}_{(2a,0),(0,0,1,0,0)} \), which is present in every concrete instance of the parametric reachability graph (there have been no retransmissions).

**timeout_retrans** is also enabled by \( M^{(MS,MR)}_{(2a,0),(0,0,1,0,0)} \) provided \( MR > 0 \).

**send_ack** is enabled by \( M^{(MS,MR)}_{(3b,0),(0,0,0,0,0)} \), which is present in every concrete instance of the parametric reachability graph (there have been no retransmissions).

**receive_ack** and **ack_loss** are enabled by \( M^{(MS,MR)}_{(2b,0),(0,0,0,1,0)} \), which is present in every concrete instance of the parametric reachability graph (there have been no retransmissions).
9.5. Channel Bounds

Table 9.1: Determining the Message Channel Bound.

<table>
<thead>
<tr>
<th>Row in Table 8.1</th>
<th>Maximum $m_o$</th>
<th>Maximum $m_n$</th>
<th>Maximum $m_o + m_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$MR$</td>
<td>0</td>
<td>$MR$</td>
</tr>
<tr>
<td>2</td>
<td>$MR$</td>
<td>$MR + 1$</td>
<td>$2MR + 1$</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>$MR$</td>
<td>$MR$</td>
</tr>
<tr>
<td>4</td>
<td>$MR - 1$</td>
<td>$MR + 1$</td>
<td>$2MR$</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>$MR$</td>
<td>$MR$</td>
</tr>
<tr>
<td>6</td>
<td>$MR - 1$</td>
<td>0</td>
<td>$MR - 1$</td>
</tr>
</tbody>
</table>

Table 9.2: Determining the Acknowledgement Channel Bound.

<table>
<thead>
<tr>
<th>Row in Table 8.1</th>
<th>Maximum $a_o$</th>
<th>Maximum $a_n$</th>
<th>Maximum $a_o + a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$MR$</td>
<td>0</td>
<td>$MR$</td>
</tr>
<tr>
<td>2</td>
<td>$MR$</td>
<td>0</td>
<td>$MR$</td>
</tr>
<tr>
<td>3</td>
<td>$MR$</td>
<td>$MR + 1$</td>
<td>$2MR + 1$</td>
</tr>
<tr>
<td>4</td>
<td>$MR - 1$</td>
<td>0</td>
<td>$MR - 1$</td>
</tr>
<tr>
<td>5</td>
<td>$MR$</td>
<td>$MR$</td>
<td>$2MR$</td>
</tr>
<tr>
<td>6</td>
<td>$MR - 1$</td>
<td>0</td>
<td>$MR - 1$</td>
</tr>
</tbody>
</table>

receive_dup_ack is enabled by $M^{(MS,MR)}_{(1,1),(0,1),(0,0),(0,0)}$, which is present in every concrete instance of the parametric reachability graph provided $MR > 0$. It is the result of one retransmission of the first message (with sequence number 0) that has subsequently been acknowledged, hence the presence of the duplicate acknowledgement.

Thus there are no dead transitions and Property 5.3 is verified.

9.5 Channel Bounds

We need to prove that Property 5.4 of Section 5.7 holds. The upper bound on the number of messages and acknowledgements in the channels can be determined by direct inspection of Table 8.1 by taking the maximum over all 6 rows in the table of the allowable values of $m_o + m_n$ for the message channel and $a_o + a_n$ for the acknowledgement channel.

For the message channel, Table 9.1 gives the maximum allowable values for $m_o$, $m_n$, and $m_o + m_n$ for each row. As highlighted in the table, row 2 has the highest value for $m_o + m_n$ and thus the bound
Table 9.3: Determining the Bound on the Total Number of Messages and Acknowledgements.

<table>
<thead>
<tr>
<th>Row in Table 8.1</th>
<th>Maximum $mo + ao$</th>
<th>Maximum $mn + an$</th>
<th>Maximum $mo + mn + ao + an$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$MR$</td>
<td>0</td>
<td>$MR$</td>
</tr>
<tr>
<td>2</td>
<td>$MR$</td>
<td>$MR + 1$</td>
<td>$2MR + 1$</td>
</tr>
<tr>
<td>3</td>
<td>$MR$</td>
<td>$MR + 1$</td>
<td>$2MR + 1$</td>
</tr>
<tr>
<td>4</td>
<td>$MR - 1$</td>
<td>$MR + 1$</td>
<td>$2MR$</td>
</tr>
<tr>
<td>5</td>
<td>$MR$</td>
<td>$MR$</td>
<td>$2MR$</td>
</tr>
<tr>
<td>6</td>
<td>$MR - 1$</td>
<td>0</td>
<td>$MR - 1$</td>
</tr>
</tbody>
</table>

on the message channel is $2MR + 1$, for any $MS \in \mathbb{N}^+$ and $MR \in \mathbb{N}$.

For the acknowledgement channel, Table 9.2 gives the maximum allowable values for $ao$, $an$, and $ao + an$ for each row. As highlighted in the table, row 3 has the highest value for $ao + an$ and thus the bound on the acknowledgement channel is also $2MR + 1$, for any $MS \in \mathbb{N}^+$ and $MR \in \mathbb{N}$.

The two bounds on the number of messages and the number of acknowledgements mean that Property 5.4 is verified. This confirms our initial proofs in [16] based on the CPN model. Furthermore, the bound on the total number of messages and acknowledgements can be determined in the same way, by maximising $mo + mn + ao + an$ in each row of Table 8.1. The row with the highest value of $mo + mn + ao + an$ is thus the upper bound. Table 9.3 gives the maximum values of $mo + ao$, $mn + an$ and $mo + ao + mn + an$ for each row of Table 8.1. As highlighted in the table, rows 2 and 3 have the highest value of $mo + mn + ao + an$ and thus the bound on the number of messages and acknowledgements in the system simultaneously is also $2MR + 1$ for any $MS \in \mathbb{N}^+$ and $MR \in \mathbb{N}$.

### 9.6 Concluding Remarks

In this chapter, the properties for verification from Section 5.7 have been verified from the algebraic expressions of the parametric reachability graph. Hence, the properties of absence of unexpected dead markings, absence of livelock, absence of unexpected dead transitions, and bounded message and acknowledgement channels have been verified for all instantiations of the SWP CPN. However, Property 5.5 has not been verified. The verification of Property 5.5, conformance to the service language, is the subject of the next chapter.
Chapter 10

Parametric Language Analysis

The next step in the parametric verification methodology presented in Chapter 3 is to consider conformance of the SWP to the Stop-and-Wait property of alternating Send and Receive events. This corresponds to Property 5.5 from Section 5.7.

Recall from Chapter 3 that 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 in Section 5.2 the set of service primitives was defined as \( SP = \{ \text{Send}, \text{Receive} \} \) and the Stop-and-Wait Service language was given in Definition 5.1 as \( L_S = (\text{Send } \text{Receive})^* \text{Send}^\dagger \) where \( \text{Send}^\dagger \) represents 0 or 1 repetitions of the Send primitive. Recall that this is because when the sender has reached the upper bound on the number of retransmissions (MaxRetrans) and all messages and acknowledgements have been lost, then the system will halt, so that the last Send event will not be followed by a corresponding Receive event. (As discussed in Chapter 5, a management entity should notify the user that the SWP has failed, but this is not usual to include this in the SWP.)

In this chapter the protocol language of the SWP is derived from the parametric RG of Theorem 8.1, 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 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 10.1.** The Stop-and-Wait protocol, as defined by the CPN in Figs. 5.5 and 5.6 conforms to the Stop-and-Wait service of alternating send and receive events, i.e. \( L_P = L_S = (\text{Send } \text{Receive})^* \text{Send}^\dagger \), for all values of the MaxSeqNo and MaxRetrans parameters.
10.1. Obtaining the Parametric Protocol Language

This theorem is proved progressively throughout this chapter, and is structured as follows. We begin in Section 10.1 by obtaining a parametric Finite State Automaton from $RG_{(MS,MR)}$ that represents our protocol language. We then calculate a parametric $\varepsilon$-closure of selected parametric markings in Section 10.2, for use in the parametric determinisation procedure combining $\varepsilon$-removal with lazy subset evaluation, which is undertaken in Section 10.3. Finally, in Section 10.4, 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.

10.1 Obtaining the Parametric 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, 108] can be used to obtain the minimal deterministic FSA of the protocol language.

In accordance with the methodology presented in [17] (see Chapter 3) we begin by defining a mapping from the binding elements of $CPN_{(MS,MR)}$ to service primitives or $\varepsilon$:

**Definition 10.1** (Mapping from Binding Elements to Service Primitives).

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

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

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

$$Prim((t, b)) = \begin{cases} 
\text{Send,} & \text{if } (t, b) = \text{send\_mess } <\text{queue} = [(i \cap MS 1)^x], sn = i>, \\
\text{Receive,} & \text{if } (t, b) = \text{receive\_mess } <\text{queue} = [i^x], sn = i, rn = i>, \\
\varepsilon, & \text{otherwise.}
\end{cases}$$

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 $\varepsilon$.

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. $M_0 = M_{(MS,MR)}^{(1,0),(0,0,0,0,0)}$. 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 $M^{(MS,MR)}_{(1,i),(0,0,0,0)}$ for all $0 \leq i \leq MS$ (incorporating
the initial marking). We also include the dead markings of $RG^{(MS,MR)}_{(MS,MR)}$ in the set of halt states, i.e.
$M^{(MS,MR)}_{(2a,i),(0,0,0,0,MR)}$ and $M^{(MS,MR)}_{(2b,i),(0,0,0,0,MR)}$ for all $0 \leq i \leq MS$, from Section 9.2. 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 $RG^{(MS,MR)}_{(MS,MR)}$:

**Definition 10.2 (FSA$_{RG^{(MS,MR)}_{(MS,MR)}}$)**

The FSA associated with $RG^{(MS,MR)}_{(MS,MR)} = (V^{(MS,MR)}, A^{(MS,MR)})$ of $CPN^{(MS,MR)}$, with initial marking
$M_0$, is $FSA_{RG^{(MS,MR)}_{(MS,MR)}} = (V^{(MS,MR)}, SP, \Delta^{(MS,MR)}, M_0, F^{(MS,MR)})$ where

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

The states of $FSA_{RG^{(MS,MR)}_{(MS,MR)}}$ are the nodes of $RG^{(MS,MR)}_{(MS,MR)}$, given by Table 8.1. The arcs of
$FSA_{RG^{(MS,MR)}_{(MS,MR)}}$ are given by Tables 8.2 to 8.7 by applying Prim to each binding element (arc label). The arcs defined by row 1 of Table 8.2 and row 1 of Table 8.7 correspond to binding elements that map to the primitive Send. The arcs defined by row 5 of Table 8.3 correspond to binding elements that map to the primitive Receive. All other arcs have binding elements that map to $\epsilon$.

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

### 10.2 Epsilon Closures

Before proceeding with FSA reduction we calculate symbolically the $\epsilon$-closure of all markings in
$V^{(MS,MR)}_{(2a,i)}$ and $V^{(MS,MR)}_{(3b,i)}$ of Table 8.1. As will become evident, we do not need to determine sym-

bolic expressions for the $\epsilon$-closure of markings in $V^{(MS,MR)}_{(1,i)}$, $V^{(MS,MR)}_{(2b,i)}$, $V^{(MS,MR)}_{(3a,i)}$ or $V^{(MS,MR)}_{(4,i)}$. This is due to the lazy subset evaluation technique for determinisation [72], discussed in Section 2.2.4 and used in Section 10.3.
To calculate the \( \epsilon \)-closure of a marking, \( M \), we use the following method. The first step is to explore the RG along arcs whose labels map to \( \epsilon \) via \( \text{Prim} \) to create a spanning of markings reachable by one or more \( \epsilon \) moves. This creates a subgraph of \( FSA_{RG(MS,MR)} \), whose edges are \( \epsilon \) moves and \( M \) is the origin. The second step is to show that all markings reachable by 0 or more \( \epsilon \) moves are actually covered by the subgraph we created, so that the set of nodes spanned by this subgraph is actually the \( \epsilon \)-closure of \( M \). This is similar in style to the proof of the Successor Lemma, Lemma 8.4.

As a final step, we determine the set of outgoing edges from the markings in the \( \epsilon \)-closure that are not labelled by \( \epsilon \), i.e. are labelled by service primitives. These sets of edges follow directly from the proofs of the \( \epsilon \)-closures and the relevant arc tables, and are thus correct by construction. These are used in the determinisation procedure in Section 10.3.

### 10.2.1 The \( \epsilon \)-closure of Nodes in \( V'(MS,MR)_{(2a,i)} \)

Recall from row 2 of Table 8.1 that \( V'(MS,MR)_{(2a,i)} = \{ M'_{(2a,i),(mo,ao,mn,0,ret)} \mid 0 \leq mo + ao \leq MR, 0 \leq \text{ret} \leq MR, 0 \leq mn \leq \text{ret} + 1 \} \). Table 8.3 defines all outgoing arcs from markings in \( V'(MS,MR)_{(2a,i)} \).

Row 5 defines arcs whose action maps to the service primitive \( \text{Receive} \). The remaining rows define arcs whose actions map to \( \epsilon \). Note that although rows 4 and 5 both correspond to an occurrence of the \( \text{receive,mess} \) transition, row 4 corresponds to reception of an old duplicate and thus maps to an \( \epsilon \) move, unlike the arcs defined by row 5.

We now begin exploring from a marking, \( M'_{(2a,i),(mo,ao,mn,0,ret)} \) \( \in V'(MS,MR)_{(2a,i)} \). From Lemma 6.4, transition \( \text{receive,mess} \) (row 4 of Table 8.3) is enabled by any marking \( M'_{(2a,i),(mo,ao,mn,0,ret)} \) if \( mo > 0 \). This leads to a marking \( M'_{(3a,i),(mo-1,ao,mn,0,ret)} \) \( \in V'(MS,MR)_{(3a,i)} \). From this marking, \( \text{send,ack} \) (row 6 of Table 8.5) is always enabled, leading to a marking \( M'_{(2a,i),(mo-1,ao+1,mn,0,ret)} \) \( \in V'(MS,MR)_{(2a,i)} \). 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 \( M'_{(2a,i),(mo,ao,mn,0,ret)} \) by the occurrence of 0 or more \( \epsilon \) transitions:

\[
V_a = \{ M'_{(2a,i),(mo-x,ao+x,mn,0,ret)} \mid 0 \leq x \leq mo \} = \{ M'_{(2a,i),(mo',ao',mn,0,ret)} \mid mo' = mo - x, ao' = ao + x, 0 \leq x \leq mo \} \tag{10.1}
\]

and

\[
V_b = \{ M'_{(3a,i),(mo-1-x,ao+x,mn,0,ret)} \mid 0 \leq x \leq mo - 1, mo > 0 \} = \{ M'_{(3a,i),(mo',ao',mn,0,ret)} \mid mo' = mo - 1 - x, ao' = ao + x, 0 \leq x \leq mo - 1, mo > 0 \} \tag{10.2}
\]
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. Just as in the proofs of the sublemmas of Lemma 8.1 (the Spanning Lemma), we omit this detail from the name of the sets for notational simplicity.

Equation (10.1) 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 (10.2) in a similar way, resulting in the following expressions for \( V_a \) and \( V_b' \):

\[
V_a = \{ M_{(2a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \mid mo' + ao' = mo + ao, 0 \leq mo' \leq mo \} \tag{10.3}
\]

and

\[
V_b = \{ M_{(3a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \mid mo' + ao' = mo + ao - 1, 0 \leq mo' \leq mo - 1, mo > 0 \} \tag{10.4}
\]

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. 10.1. 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
\text{mess} followed by send
\text{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
\text{mess} occurs, the number of old messages decreases by one, and every time send
\text{ack} occurs, the number of old acknowledgements increases by one.

The transition \text{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 8.3 and Table 8.5 respectively). 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 8.3 and Table 8.5, firing \text{timeout_retrans} from each marking in \( V_a \) and \( V_b \) results in the following sets of markings reachable by 0 or more \( \epsilon \) moves:

\[
V_a' = \{ M_{(2a,i),(mo',ao',mn+y,0,ret+y)}^{(MS,MR)} \mid M_{(2a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \in V_a, 0 \leq y \leq MR - ret \} \tag{10.5}
\]

\[
V_b' = \{ M_{(3a,i),(mo',ao',mn+y,0,ret+y)}^{(MS,MR)} \mid M_{(3a,i),(mo',ao',mn,0,ret)}^{(MS,MR)} \in V_b, 0 \leq y \leq MR - ret \} \tag{10.6}
\]
10.2. Epsilon Closures

Figure 10.1: The markings, represented by \( m_o \), \( a_o \) and \( m_n \), reachable from \( M_{(2a,1)}^{(MS,MR)}(m_o,a_o,m_n,0,\text{ret}) \) through repeated successive firings of \texttt{receive mess} followed by \texttt{send ack}. 
10.2. Epsilon Closures

Substituting (10.3) and (10.4) into (10.5) and (10.6) respectively gives:

\[ V'_a = \{ M^{(MS,MR)}_{(2a,i),(mo',ao',mn',0,ret')} \mid mo' + ao' = mo + ao, 0 \leq mo' \leq mo, mn' = mn + y, \]
\[ \text{ret}' = \text{ret} + y, 0 \leq y \leq MR - \text{ret} \} \] (10.7)

\[ V'_b = \{ M^{(MS,MR)}_{(3a,i),(mo',ao',mn',0,ret')} \mid mo' + ao' = mo + ao - 1, 0 \leq mo' \leq mo - 1, mn' = mn + y, \]
\[ \text{ret}' = \text{ret} + y, 0 \leq y \leq MR - \text{ret}, mo > 0 \} \] (10.8)

Substituting the expression \( y = \text{ret}' - \text{ret} \) into (10.7) and (10.8) to eliminate \( y \) gives:

\[ V'_a = \{ M^{(MS,MR)}_{(2a,i),(mo',ao',mn',0,ret')} \mid mo' + ao' = mo + ao, 0 \leq mo' \leq mo, mn' = mn + \text{ret}' - \text{ret}, \]
\[ \text{ret} \leq \text{ret}' \leq MR \} \] (10.9)

\[ V'_b = \{ M^{(MS,MR)}_{(3a,i),(mo',ao',mn',0,ret')} \mid mo' + ao' = mo + ao - 1, 0 \leq mo' \leq mo - 1, \]
\[ mn' = mn + \text{ret}' - \text{ret}, \text{ret} \leq \text{ret}' \leq MR, mo > 0 \} \] (10.10)

The sets \( V'_a \) and \( V'_b \) are illustrated in the context of \( V^{(MS,MR)}_{(2a,i)} \) and \( V^{(MS,MR)}_{(3a,i)} \) by Fig. 10.2. From each marking identified in \( V_a \) and \( V_b \), as illustrated in Fig. 10.1, 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 \( mn' = mn + MR - \text{ret} \), at which point it is no longer enabled.

By Lemmas 6.8 and 6.9, transitions mess_joss (rows 2 and 3 of Table 8.3) and ack_joss (row 6 of Table 8.3) 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 8.5 allow the same for \( V'_b \). Applying Definition 7.11 to (10.9) and (10.10) to obtain the downward-closed sets of all markings in \( V'_a \) and \( V'_b \) gives:

\[ V''_a = DC(V'_a) \]
\[ = \{ M^{(MS,MR)}_{(2a,i),(mo'',ao'',mn'',0,ret')} \mid M^{(MS,MR)}_{(2a,i),(mo',ao',mn',0,ret')} \in V'_a, mo'' \leq mo', ao'' \leq ao', mn'' \leq mn' \} \]
\[ = \{ M^{(MS,MR)}_{(2a,i),(mo'',ao'',mn'',0,ret')} \mid mo'' \leq mo', ao'' \leq ao', mn'' \leq mn', mo' + ao' = mo + ao, \]
\[ 0 \leq mo' \leq mo, mn' = mn + \text{ret}' - \text{ret}, \text{ret} \leq \text{ret}' \leq MR \} \] (10.11)

\[ V''_b = DC(V'_b) \]
\[ = \{ M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn'',0,ret')} \mid M^{(MS,MR)}_{(3a,i),(mo',ao',mn',0,ret')} \in V'_b, mo'' \leq mo', ao'' \leq ao', mn'' \leq mn' \} \]
\[ = \{ M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn'',0,ret')} \mid mo'' \leq mo', ao'' \leq ao', mn'' \leq mn', mo' + ao' = mo + ao - 1, \]
\[ 0 \leq mo' \leq mo - 1, mn' = mn + \text{ret}' - \text{ret}, \text{ret} \leq \text{ret}' \leq MR, mo > 0 \} \] (10.12)
10.2. Epsilon Closures

Figure 10.2: The markings from \( V^{(MS, MR)} \) and \( V^{(MS, MR)} \), represented by \( mo \), \( ao \) and \( mn \), reachable from the markings in Fig. 10.1 through repeated firing of timeout/retrans.
The variables \( mo', ao' \) and \( mn' \) can be eliminated in (10.11) and (10.12) by substituting the inequalities \( mo'' \leq mo', ao'' \leq ao' \) and \( mn'' \leq mn' \) wherever \( mo', ao' \) and \( mn' \) appear, giving:

\[
V''_a = \{ M^{(MS, MR)}_{(2a,i),(mo'', ao'', mn'', 0, ret')} \mid mo'' + ao'' \leq mo + ao, 0 \leq mo'' \leq mo, mn'' \leq mn + ret' - ret, \cr ret \leq ret' \leq MR \} \tag{10.13}
\]

\[
V''_b = \{ M^{(MS, MR)}_{(3a,i),(mo'', ao'', mn'', 0, ret')} \mid mo'' + ao'' \leq mo + ao - 1, 0 \leq mo'' \leq mo - 1, \cr mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR, mo > 0 \} \tag{10.14}
\]

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. 10.1 and 10.2, as it becomes a complex three-dimensional lattice of markings. Instead, we return to the more abstract view from Chapter 8 and illustrate the coverage of \( V''_a \) and \( V''_b \) over \( V^{(MS, MR)}_i \) in Fig. 10.3. The shaded region in \( V^{(MS, MR)}_{(2a,i)} \) represents \( V''_a \) and the shaded region in \( V^{(MS, MR)}_{(3a,i)} \) represents \( V''_b \). Naturally, the coverage will depend on which specific initial marking, \( M^{(MS, MR)}_{(2a,i),(mo, ao, mn, 0, ret)} \) (the large red circle), is chosen. For example, if \( M^{(MS, MR)}_{(2a,i),(MR, 0, 0, 0)} \) is chosen as the initial marking, then all of \( V^{(MS, MR)}_{(2a,i)} \) is covered by \( V''_a \).

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

**Lemma 10.1.** *Any marking \( M' \in V^{(MS, MR)} \) reachable via an \( \epsilon \) 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 \( M^{(MS, MR)}_{(2a,i),(mo', ao', mn', 0, ret')} \) in \( V''_a \) at most 6 outgoing \( \epsilon \) moves are defined by Table 8.3 and mapping \( Prim \). They are treated systematically below. The following relies heavily on Equations (10.13) and (10.14) and references are made when appropriate.

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

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

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

All three of these resulting markings are identical to their source marking, \( M^{(MS, MR)}_{(2a,i),(mo', ao'', mn'', 0, ret')} \) 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 7.11) all three resulting markings are also elements of the downward-closed set, \( V_0^\epsilon_a \).

The remaining two outgoing \( \epsilon \) moves require more careful treatment:

**timeout_retrans** (row 1) is enabled by \( M^{(MS,MR)}_{(2a,i), (mo^\epsilon, ao^\epsilon, mn^\epsilon, 0, ret^\epsilon)} \) \( \in V_0^\epsilon_a \) if \( ret^\epsilon < MR \). Its occurrence results in a marking \( M^{(MS,MR)}_{(2a,i), (mo^\epsilon, ao^\epsilon, mn^\epsilon + 1, 0, ret^\epsilon + 1)} \). From the inequalities in Equation (10.13), \( (mn^\epsilon + 1) \leq mn + (ret^\epsilon + 1) - ret \), and \( ret < (ret^\epsilon + 1) \leq MR \) because of the restriction...
that \( ret' < MR \). These do not violate the inequalities given in Equation (10.13) when substituting \((mn'' + 1)\) and \((ret' + 1)\) for \(mn''\) and \(ret'\) respectively into Equation (10.13). Hence \(M^{(MS,MR)}_{(2a,i),(mo'',ao'',mn''+1,0,ret'+1)} \in V''_a\).

**receive**_{mess} (row 4) is enabled by \(M^{(MS,MR)}_{(2a,i),(mo'',ao'',mn'',0,ret')} \in V''_a\) if \(mo'' > 0\). Its occurrence results in a marking \(M^{(MS,MR)}_{(2a,i),(mo''-1,ao'',mn'',0,ret')}\). From the inequalities in Equation (10.13), 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 (10.14) when substituting \((mo'' - 1)\) for \(mo''\) in Equation (10.14). By inspection, the other inequalities from Equation (10.13) are the same as corresponding inequalities in Equation (10.14). Hence \(M^{(MS,MR)}_{(2a,i),(mo''-1,ao'',mn'',0,ret')} \in V''_b\).

For \(M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn'',0,ret')} \in V''_b\) at most 6 outgoing \(\epsilon\) moves are defined by Table 8.5 and mapping Prim. They are treated systematically below. Again, the following relies heavily on Equations (10.13) and (10.14) and references are made when appropriate.

**mess**_{loss_old} (row 2) is enabled by \(M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn'',0,ret')} \in V''_b\) if \(mo'' > 0\). Its occurrence leads to a marking \(M^{(MS,MR)}_{(3a,i),(mo''-1,ao'',mn'',0,ret')}\).

**mess**_{loss_new} (row 3) is enabled by \(M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn'',0,ret')} \in V''_b\) if \(mn'' > 0\). Its occurrence leads to a marking \(M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn''-1,0,ret')}\).

**ack**_{loss_old}, **receive**_{dup ack} (rows 4 and 5) are enabled by \(M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn'',0,ret')} \in V''_b\) if \(ao'' > 0\). Its occurrence of either leads to a marking \(M^{(MS,MR)}_{(3a,i),(mo'',ao''-1,mn'',0,ret')}\).

All three of these resulting markings are identical to their source marking, \(M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn'',0,ret')} \in V''_b\), 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 7.11), all three resulting markings are also elements of \(V''_b\).

The remaining two outgoing \(\epsilon\) moves require more careful treatment:

**timeout**_{retrans} (row 1) is enabled by \(M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn'',0,ret')} \in V''_b\) if \(ret' < MR\). Its occurrence results in a marking \(M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn''+1,0,ret'+1)}\). From the inequalities in Equation (10.14), \((mn'' + 1) \leq mn + (ret' + 1) - ret\), and \(ret < (ret' + 1) \leq MR\) because of the restriction that \(ret' < MR\). These do not violate the inequalities given in Equation (10.14) when substituting \((mn'' + 1)\) and \((ret' + 1)\) for \(mn''\) and \(ret'\) into Equation (10.14). Hence \(M^{(MS,MR)}_{(3a,i),(mo'',ao'',mn''+1,0,ret'+1)} \in V''_b\).
Corollary 10.1. The \( \epsilon \)-closure of a marking \( M^{(MS,MR)}_{(2a,i),(mo,ao,mm,0,ret)} \) is given by the union of the parameterised sets \( V''_a \) and \( V''_b \) defined by Equations (10.13) and (10.14) respectively, i.e. (replacing double primes with single primes):

\[
\text{Closure}(M^{(MS,MR)}_{(2a,i),(mo,ao,mm,0,ret)}) = \{ M^{(MS,MR)}_{(2a,i),(mo',ao',mm',0,ret')} \mid mo' + ao' \leq mo + ao, \\
0 \leq mo' \leq mo, mm' \leq mn + ret' - ret, ret \leq ret' \leq MR \}
\]

\[
\cup \{ M^{(MS,MR)}_{(3a,i),(mo',ao',mm',0,ret')} \mid mo' + ao' \leq mo + ao - 1, 0 \leq mo' \leq mo - 1, \\
mm' \leq mn + ret' - ret, ret \leq ret' \leq MR, mo > 0 \} \quad (10.15)
\]

The only transition enabled by the markings in \( \text{Closure}(M^{(MS,MR)}_{(2a,i),(mo,ao,mm,0,ret)}) \) that does not map to \( \epsilon \) is given by Row 5 of Table 8.3 and maps to the Receive service primitive. From Row 5 of Table 8.3, the corresponding set of non-\( \epsilon \)-labelled outgoing edges from all markings in the above closure is given by:

\[
\text{OutEdges}(\text{Closure}(M^{(MS,MR)}_{(2a,i),(mo,ao,mm,0,ret)})) = \{ M^{(MS,MR)}_{(2a,i),(0,ao',mm',0,ret')} \mid M^{(MS,MR)}_{(3b,i),(0,ao',mm'-1,0,ret')} \in V''_a, mn' > 0 \} \quad (10.16)
\]

Substituting \( V''_a \) from Equation (10.13) into Equation (10.16) we obtain:

\[
\text{OutEdges}(\text{Closure}(M^{(MS,MR)}_{(2a,i),(mo,ao,mm,0,ret)})) = \{ M^{(MS,MR)}_{(2a,i),(0,ao',mm',0,ret')} \mid ao' \leq mo + ao, mn' \leq mn + ret' - ret, ret \leq ret' \leq MR, mn' > 0 \}
\]

\[
(10.17)
\]

10.2.2 The \( \epsilon \)-closure of Nodes in \( V^{(MS,MR)}_{(3b,i)} \)

Recall from row 5 of Table 8.1 that \( V^{(MS,MR)}_{(3b,i)} = \{ M^{(MS,MR)}_{(3b,i),(0,ao,mm,an,ret)} \mid 0 \leq ao \leq MR, 0 \leq ret \leq MR, 0 \leq mn + an \leq ret \} \). Table 8.6 defines all outgoing arcs from markings in \( V^{(MS,MR)}_{(3b,i)} \). All rows define arcs whose action maps to \( \epsilon \).
10.2. Epsilon Closures

The transition `timeout_retrans` (row 1 of Table 8.6) is enabled by all markings \(M^{(MS,MR)}_{(3b,i),(0,ao,mm,an,ret)} \in V^{(MS,MR)}_{(3b,i)}\) in which \(ret < MR\). Its occurrence leads to a marking \(M^{(MS,MR)}_{(3b,i),(0,ao,mm+1,an,ret+1)}\). 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 \(M^{(MS,MR)}_{(3b,i),(0,ao,mm,an,ret)}\), i.e. until the maximum number of retransmissions is reached. The occurrence of `timeout_retrans` from \(M^{(MS,MR)}_{(3b,i),(0,ao,mm,an,ret)} \in V^{(MS,MR)}_{(3b,i)}\), 0 to \(MR - ret\) number of times, results in the set of markings, \(V_c\), the elements of which are reachable from \(M^{(MS,MR)}_{(3b,i),(0,ao,mm,an,ret)}\) by 0 or more \(\epsilon\) moves:

\[
V_c = \{M^{(MS,MR)}_{(3b,i),(0,ao,mm+x,an,ret+x)} \mid 0 \leq x \leq MR - ret\}
= \{M^{(MS,MR)}_{(3b,i),(0,ao,mm',an,ret')} \mid mn' = mn + x, ret' = ret + x, 0 \leq x \leq MR - ret\} \tag{10.18}
\]

Equation (10.18) can be simplified by eliminating \(x\), by substituting \(x = ret' - ret\):

\[
V_c = \{M^{(MS,MR)}_{(3b,i),(0,ao,mm',an,ret')} \mid mn' = mn + ret' - ret, ret \leq ret' \leq MR\} \tag{10.19}
\]

Transition `send_ack` (row 7 of Table 8.6) is enabled by any marking \(M^{(MS,MR)}_{(3b,i),(0,ao,mm',an,ret')} \in V_c\). This leads to a marking \(M^{(MS,MR)}_{(2b,i),(0,ao,mm',an+1,ret')} \in V^{(MS,MR)}_{(2b,i)}\). From this marking, `receive_mess` (row 3 of Table 8.4) is enabled, and can occur, provided \(mn' > 0\), leading to a marking \(M^{(MS,MR)}_{(3b,i),(0,ao,mm'-1,an+1,ret')}\). 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 \(M^{(MS,MR)}_{(3b,i),(0,ao,mm,an,ret)}\) by 0 or more \(\epsilon\) transitions:

\[
V_d = \{M^{(MS,MR)}_{(3b,i),(0,ao,mm'-y,an+y,ret')} \mid M^{(MS,MR)}_{(3b,i),(0,ao,mm',an,ret')} \in V_c, 0 \leq y \leq mn'\} \tag{10.20}
\]

\[
V_e = \{M^{(MS,MR)}_{(2b,i),(0,ao,mm'-y,an+1+y,ret')} \mid M^{(MS,MR)}_{(3b,i),(0,ao,mm',an,ret')} \in V_c, 0 \leq y \leq mn'\} \tag{10.21}
\]

Using Equation (10.19) in Equations (10.20) and (10.21) gives:

\[
V_d = \{M^{(MS,MR)}_{(3b,i),(0,ao,mm'',an',ret')} \mid mn'' = mn' - y, an' = an + y, mn'' = mn + ret' - ret, ret \leq ret' \leq MR, 0 \leq y \leq mn'\}
= \{M^{(MS,MR)}_{(3b,i),(0,ao,mm'',an',ret')} \mid mn'' = mn + ret' - ret - y, an' = an + y, ret \leq ret' \leq MR, 0 \leq y \leq mn + ret' - ret\} \tag{10.22}
\]
10.2. Epsilon Closures

\[ V_e = \{ M^{(MS, MR)}_{(2b, i), (0, ao, mn''', an', ret')} : mn'' = mn' - y, an' = an + 1 + y, mn' = mn + ret' - ret, \]
\[ \text{ret} \leq \text{ret}' \leq MR, 0 \leq y \leq mn' \}\]

\[ = \{ M^{(MS, MR)}_{(2b, i), (0, ao, mn''', an', ret')} : mn'' = mn + ret' - ret - y, an' = an + 1 + y, \text{ret} \leq \text{ret}' \leq MR, \]
\[ 0 \leq y \leq mn + ret' - ret \} \quad (10.23) \]

Equation (10.22) 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 (10.23) in a similar way, resulting in the following expressions for \( V_d \) and \( V_e \):

\[ V_d = \{ M^{(MS, MR)}_{(3b, i), (0, ao, mn''', an', ret')} : mn'' + an' = mn + an + ret' - ret, 0 \leq mn'' \leq mn + ret' - ret, \]
\[ \text{ret} \leq \text{ret}' \leq MR \} \quad (10.24) \]

\[ V_e = \{ M^{(MS, MR)}_{(2b, i), (0, ao, mn''', an', ret')} : mn'' + an' = mn + an + ret' - ret + 1, 0 \leq mn'' \leq mn + ret' - ret, \]
\[ \text{ret} \leq \text{ret}' \leq MR \} \quad (10.25) \]

Transition mess.loss (row 2 of Table 8.6) and ack.loss (rows 3 and 4 of Table 8.6) 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 8.4 allow the same for \( V_e \). Applying Definition 7.11 to (10.24) and (10.25) to obtain the downward-closed sets of all markings in \( V_d \) and \( V_e \) gives:

\[ V_d' = DC(V_d) \]
\[ = \{ M^{(MS, MR)}_{(3b, i), (0, ao', mn''', an', ret')} : M^{(MS, MR)}_{(3b, i), (0, ao, mn''', an', ret')} \in V_d, ao' \leq ao, mn''' \leq mn'', an'' \leq an' \} \]
\[ = \{ M^{(MS, MR)}_{(3b, i), (0, ao', mn''', an', ret')} : ao' \leq ao, mn''' \leq mn'', an'' \leq an', mn'' + an' = mn + an + \]
\[ + ret' - ret, 0 \leq mn'' \leq mn + ret' - ret, \text{ret} \leq \text{ret}' \leq MR \} \quad (10.26) \]

\[ V_e' = DC(V_e) \]
\[ = \{ M^{(MS, MR)}_{(2b, i), (0, ao', mn''', an', ret')} : M^{(MS, MR)}_{(2b, i), (0, ao, mn''', an', ret')} \in V_e, ao' \leq ao, mn''' \leq mn'', an'' \leq an' \} \]
\[ = \{ M^{(MS, MR)}_{(2b, i), (0, ao', mn''', an', ret')} : ao' \leq ao, mn''' \leq mn'', an'' \leq an', mn'' + an' = mn + an + \]
\[ + ret' - ret + 1, 0 \leq mn'' \leq mn + ret' - ret, \text{ret} \leq \text{ret}' \leq MR \} \quad (10.27) \]
10.2. Epsilon Closures

The variables $mn''$ and $an'$ can be eliminated from Equations (10.26) and (10.27) by substituting the inequalities $mn'' \leq mn''$ and $an'' \leq an'$ wherever $mn''$ and $an'$ appear, giving:

$$V'_d = \{ M^{(MS,MR)}_{(3h,i)},(0,ao',mn'',an'',ret') \mid ao' \leq ao, mn'' + an'' \leq mn + an + ret' - ret, \\
0 \leq mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR \} \quad (10.28)$$

$$V'_e = \{ M^{(MS,MR)}_{(2h,i)},(0,ao',mn'',an'',ret') \mid ao' \leq ao, mn'' + an'' \leq mn + an + ret' - ret + 1, \\
0 \leq mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR \} \quad (10.29)$$

Transition receive_ack (row 6 of Table 8.6) is enabled and can occur from any marking $M^{(MS,MR)}_{(3h,i),}(0,ao',mn'',an'',ret') \in V'_d$ in which $ao' = 0$ and $an'' > 0$, leading to a marking $M^{(MS,MR)}_{(4,i\oplus_M S_1),}(mn'',an''-1,0,0,0) \in V^{(MS,MR)}_{(4,i\oplus_M S_1)}$. Also, receive_ack (row 7 of Table 8.4) is enabled by any marking $M^{(MS,MR)}_{(2h,i),}(0,ao',mn'',an'',ret') \in V'_e$ in which $ao' = 0$ and $an'' > 0$, leading to a marking $M^{(MS,MR)}_{(1,i\oplus_M S_1),}(mn'',an''-1,0,0,0) \in V^{(MS,MR)}_{(1,i\oplus_M S_1)}$. 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:

$$V_f = \{ M^{(MS,MR)}_{(4,i\oplus_M S_1),}(mn'',an''-1,0,0,0) \mid M^{(MS,MR)}_{(3h,i),}(0,0,mn'',an'',ret') \in V'_d, an'' > 0 \} \quad (10.30)$$

$$V_g = \{ M^{(MS,MR)}_{(1,i\oplus_M S_1),}(mn'',an''-1,0,0,0) \mid M^{(MS,MR)}_{(2h,i),}(0,0,mn'',an'',ret') \in V'_e, an'' > 0 \} \quad (10.31)$$

Using Equations (10.28) and (10.29) into (10.30) and (10.31) respectively gives:

$$V_f = \{ M^{(MS,MR)}_{(4,i\oplus_M S_1),}(mn'',an''-1,0,0,0) \mid an'' = an'' - 1, 0 \leq mn'' + an'' \leq mn + an + ret' - ret, \\
0 \leq mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR, an'' > 0 \} \quad (10.32)$$

$$V_g = \{ M^{(MS,MR)}_{(1,i\oplus_M S_1),}(mn'',an''-1,0,0,0) \mid an'' = an'' - 1, 0 \leq mn'' + an'' \leq mn + an + ret' - ret + 1, \\
0 \leq mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR, an'' > 0 \} \quad (10.33)$$

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

$$V_f = \{ M^{(MS,MR)}_{(4,i\oplus_M S_1),}(mn'',an''-1,0,0,0) \mid 0 \leq mn'' + an'' \leq mn + an + ret' - ret - 1, \\
0 \leq mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR \} \quad (10.34)$$

$$V_g = \{ M^{(MS,MR)}_{(1,i\oplus_M S_1),}(mn'',an''-1,0,0,0) \mid 0 \leq mn'' + an'' \leq mn + an + ret' - ret, \\
0 \leq mn'' \leq mn + ret' - ret, ret \leq ret' \leq MR \} \quad (10.35)$$
An abstract view of the coverage of $V_d'$, $V_e'$, $V_f$ and $V_g$ over $V^{(MS,MR)}_i$ and $V^{(MS,MR)}_{i+MS1}$ is given in Fig. 10.4. $V_d'$, $V_e'$, $V_f$ and $V_g$ are represented by the corresponding shaded portions of Fig. 10.4. $V_d'$ is a subset of $V^{(MS,MR)}_{(3b,i)}$, $V_e'$ is a subset of $V^{(MS,MR)}_{(2b,i)}$, $V_f$ is a subset of $V^{(MS,MR)}_{(4,i+MS1)}$, and $V_g$ is a subset of $V^{(MS,MR)}_{(1,i+MS1)}$. 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. 10.3, the extent of coverage will depend upon the initial marking, $M^{(MS,MR)}_{(3b,i),(0,ao,mm,an,ret)}$, chosen.

At this point we conjecture that $V_d' \cup V_e' \cup V_f \cup V_g$ is the $\epsilon$-closure of a marking $M^{(MS,MR)}_{(3b,i),(0,ao,mm,an,ret)}$. We must ensure that there are no additional markings reachable via $\epsilon$ moves:

**Lemma 10.2.** Any marking $M' \in V^{(MS,MR)}$ reachable via an $\epsilon$ 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 $M^{(MS,MR)}_{(3b,i),(0,ao',mm',an',ret')}$ at most 7 outgoing $\epsilon$ moves are defined by Table 8.6 and mapping Prim. They are treated systematically below.

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

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

- **ack_loss_new** (row 4) is enabled by $M^{(MS,MR)}_{(3b,i),(0,ao',mm',an',ret')}$ if $an'' > 0$. Its occurrence leads to a marking $M^{(MS,MR)}_{(3b,i),(0,ao',mm''-1,an',ret')}$.

All three of these resulting markings are identical to their source marking, $M^{(MS,MR)}_{(3b,i),(0,ao',mm',an',ret')}$ 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 7.11), all three resulting markings are also elements of $V_d'$.

The remaining outgoing $\epsilon$ moves are a little more complex. treatment:

- **timeout_retrans** (row 1) is enabled by $M^{(MS,MR)}_{(3b,i),(0,ao',mm',an',ret')}$ if $ret' < MR$. Its occurrence results in a marking $M^{(MS,MR)}_{(3b,i),(0,ao',mm''+1,an',ret'+1)}$. From the inequalities in Equation (10.28), $(mn'' + 1) + an'' \leq mn + an + (ret' + 1) - ret$, $0 < (mn'' + 1) \leq mn + (ret' + 1) - ret$, and $ret < (ret' + 1) \leq MR$ because of the restriction that $ret' < MR$. These satisfy the
10.2. Epsilon Closures

Figure 10.4: The coverage of $V_d'$, $V_e'$, $V_f$ and $V_g$ over $V_i^{(MS,MR)} \cup V_{i_{MS,1}}^{(MS,MR)}$. 

$M^{(MS,MR)}_{(1,i),(MR,0,0,0,0)}$  

$M^{(MS,MR)}_{(3b,i),(0,ao,nm,an,rel)}$  

$V_{d'}^{(MS,MR)}$  

$V_{e'}^{(MS,MR)}$  

$V_f^{(MS,MR)}$  

$V_g^{(MS,MR)}$
inequalities given in Equation (10.28) when substituting \((mn''' + 1)\) and \((ret' + 1)\) for \(mn'''\) and 
\(ret'\) respectively into Equation (10.28). Hence \(M^{(MS,MR)}_{(2b, i), (0, ao', mn''' + 1, an'', ret' + 1)} \in V'_d\). 

**receive_ack (row 6)** is enabled by 
\(M^{(MS,MR)}_{(3b, i), (0, ao', mn'', an', ret')} \in V'_d\) if \(ao' = 0\) and \(an'' > 0\). Its occurrence leads to a marking 
\(M^{(MS,MR)}_{(4, i \in MS), (mn'', an'' - 1, 0, 0, 0)}\). From the inequalities in Equation 
(10.28), \(mn'' + (an'' - 1) \leq mn + an + ret' - ret - 1\). This matches the corresponding inequality 
in Equation (10.34) when substituting \((an'' - 1)\) for \(an''\) in Equation (10.34). By inspection, the 
other inequalities from Equation (10.28) are the same as the corresponding inequalities in Equation 
(10.34). Hence 
\(M^{(MS,MR)}_{(4, i \in MS), (mn''', an'' - 1, 0, 0, 0)} \in V'_f\). 

**send_ack (row 7)** is enabled by all markings in \(V'_d\). Its occurrence leads to a marking 
\(M^{(MS,MR)}_{(2b, i), (0, ao', mn''', an', ret')} \in V'_e\) at most 7 outgoing \(\epsilon\) moves are defined by Table 8.4 and mapping 
Prim. As before, we consider them one by one. 

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

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

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

All three of these resulting markings are identical to their source marking, 
\(M^{(MS,MR)}_{(2b, i), (0, ao', mn''', an', ret')} \in V'_e\), 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 7.11), all three resulting markings are also elements of \(V'_e\). 

The remaining three outgoing \(\epsilon\) moves require more careful treatment: 

**timeout_retrans (row 1)** is enabled by 
\(M^{(MS,MR)}_{(2b, i), (0, ao', mn''', an', ret')} \in V'_e\) if \(ret' < MR\). Its occurrence 
results in a marking 
\(M^{(MS,MR)}_{(2b, i), (0, ao', mn''', an', ret') + 1}\). From the inequalities in Equation (10.29), 
\((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 (10.29) when substituting \((mn'' + 1)\) and \((ret' + 1)\) for \(mn'''\) and \(ret'\) respectively into Equation (10.29). Hence \(M^{(MS,MR)}_{(2b,i),(0,ao',mn'''+1,an',ret'+1)} \in V'_e\).

**receive_mess (row 3)** is enabled by \(M^{(MS,MR)}_{(2b,i),(0,ao',mn''',an',ret')} \in V'_e\) if \(mn''' > 0\). Its occurrence results in a marking \(M^{(MS,MR)}_{(3b,i),(0,ao',mn''',an',ret')}\). From the inequalities in Equation (10.29), \((mn''' - 1) + an'' \leq mn + an + ret' - ret\), and \(0 \leq (mn''' - 1) < mn + ret' - ret\) because of the restriction that \(mn''' > 0\). These satisfy the corresponding inequalities in Equation (10.28) when substituting \((mn''' - 1)\) for \(mn''\) in Equation (10.28). By inspection, the other two inequalities from Equation (10.29) are the same as the corresponding inequalities in Equation (10.28). Hence \(M^{(MS,MR)}_{(3b,i),(0,ao',mn''',an',ret')} \in V'_e\).

**receive_ack (row 7)** is enabled by \(M^{(MS,MR)}_{(2b,i),(0,ao',mn''',an',ret')} \in V'_e\) if \(ao' = 0\) and \(an'' > 0\). Its occurrence results in a marking \(M^{(MS,MR)}_{(1,i\oplus MS1),(mn'''',an'',0,0,0)}\). From the inequalities in Equation (10.29), \(mn''' + (an'' - 1) \leq mn + an + ret' - ret\). This matches the corresponding inequality in Equation (10.35) when substituting \((an'' - 1)\) for \(an'''\) into Equation (10.35). By inspection, the other two relevant inequalities from Equation (10.29) match the corresponding inequalities in Equation (10.35). Hence \(M^{(MS,MR)}_{(1,i\oplus MS1),(mn'''',an'',0,0,0)} \in V'_e\).

For \(M^{(MS,MR)}_{(4,i\oplus MS1),(mn''',an'',0,0,0)} \in V_f\) at most 4 outgoing \(\epsilon\) moves are defined by Table 8.7 and mapping \(Prim\), substituting \(i \oplus MS1\) for \(i\). Three of them correspond to loss events:

**mess_loss_old (row 2)** is enabled by \(M^{(MS,MR)}_{(4,i\oplus MS1),(mn''',an'',0,0,0)} \in V_f\) if \(mn''' > 0\). Its occurrence results in a marking \(M^{(MS,MR)}_{(4,i\oplus MS1),(mn''',an'',0,0,0)}\).

**ack_loss_old and receive_dup_ack (rows 3 and 4)** are enabled by \(M^{(MS,MR)}_{(4,i\oplus MS1),(mn''',an'',0,0,0)} \in V_f\) if \(an'' > 0\). Its occurrence of either leads to a marking \(M^{(MS,MR)}_{(4,i\oplus MS1),(mn''',an'',0,0,0)}\).

Both of these resulting markings are identical to \(M^{(MS,MR)}_{(4,i\oplus MS1),(mn''',an'',0,0,0)} \in V_f\) except for either one less old message or one less old acknowledgement. From Equation (10.34), 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 7.11), both of these resulting markings are also elements of \(V_f\).

The fourth outgoing \(\epsilon\) move corresponds to **send_ack**:

**send_ack (row 5)** is enabled by all markings in \(V_f\). Its occurrence results in a marking \(M^{(MS,MR)}_{(1,i\oplus MS1),(mn''',an'''+1,0,0,0)}\). From the inequalities in Equation (10.34), \(0 \leq mn''' + (an'''+1) \leq mn + an + ret' - ret\). This matches the corresponding inequality in Equation (10.35) when substituting \((an'''+1)\) for \(an'''\) in Equation (10.35). By inspection, the other two inequalities
from Equation (10.34) are the same as the corresponding inequalities in Equation (10.35). Hence

\[ M^{(MS,MR)}_{(1,i \oplus MS 1), (mn''',an''',0,0,0)} \in V_f. \]

In a nearly identical fashion to \( M^{(MS,MR)}_{(4,i \oplus MS 1), (mn''',an''',0,0,0)} \in V_g \)
also has at most 4 outgoing \( \epsilon \) moves, defined by Table 8.2 and mapping \( Prim \), substituting \( i \oplus MS 1 \) for \( i \). Three of the outgoing \( \epsilon \) 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 \( \epsilon \) moves from \( V_f \), hence are not repeated. The only other outgoing \( \epsilon \) move corresponds to receive

\textbf{ness} on row 3 of Table 8.2. This transition is enabled by \( M^{(MS,MR)}_{(1,i \oplus MS 1), (mn''',an''',0,0,0)} \in V_g \)
if \( mn''' > 0 \). Its occurrence results in a marking \( M^{(MS,MR)}_{(4,i \oplus MS 1), (mn''',an''',0,0,0)} \) . From the inequalities

in Equation (10.35), \( 0 \leq (mn''' - 1) + an''' \leq mn + an + ret' - ret - 1 \) and \( 0 \leq (mn''' - 1) < mn + ret' - ret \) because of the restriction that \( mn''' > 0 \). These satisfy the corresponding inequalities in Equation (10.34) when substituting \( (mn''' - 1) \) for \( mn''' \) in Equation (10.34). By inspection, the remaining inequality (on \( ret \)) from Equation (10.35) matches the corresponding inequality in Equation (10.34). Hence \( M^{(MS,MR)}_{(4,i \oplus MS 1), (mn''',an''',0,0,0)} \in V_f \).

All successors of all markings in \( V_d' \cup V_e' \cup V_f' \cup V_g \) reachable by an \( \epsilon \) 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 10.2.** The \( \epsilon \)-closure of \( M^{(MS,MR)}_{(3b,i), (0,ao,mn,an,ret)} \cdot \text{Closure}(M^{(MS,MR)}_{(3b,i), (0,ao,mn,an,ret)}) \), is given by

the union of the parameterised sets \( V_d' \cup V_e' \cup V_f' \cup V_g \) defined by Equations (10.28), (10.29), (10.34) and (10.35) respectively, i.e. (replacing double and triple primes with single primes):

\[
\text{Closure}(M^{(MS,MR)}_{(3b,i), (0,ao,mn,an,ret)}) = \{ M^{(MS,MR)}_{(3b,i), (0,ao',mn',an',ret')} \mid ao' \leq ao, \\
\quad mn' + an' \leq mn + an + ret' - ret, 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR \}
\]

\[
\cup \{ M^{(MS,MR)}_{(2b,i), (0,ao',mn',an',ret')} \mid ao' \leq ao, mn' + an' \leq mn + an + ret' - ret + 1, \\
\quad 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR \}
\]

\[
\cup \{ M^{(MS,MR)}_{(4,i \oplus MS 1), (mn',an',0,0,0)} \mid mn' + an' \leq mn + an + ret' - ret - 1, \\
\quad 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR \}
\]

\[
\cup \{ M^{(MS,MR)}_{(1,i \oplus MS 1), (mn',an',0,0,0)} \mid mn' + an' \leq mn + an + ret' - ret, \\
\quad 0 \leq mn' \leq mn + ret' - ret, ret \leq ret' \leq MR \} \quad (10.36)
\]

The only transitions enabled by the markings in \( \text{Closure}(M^{(MS,MR)}_{(3b,i), (0,ao,mn,an,ret)}) \) that do not map to \( \epsilon \) are given by Row 1 of Table 8.2 and Row 1 of Table 8.7. They both map to the Send service primitive. From Row 1 of Table 8.2 and Row 1 of Table 8.7 the corresponding set of non-\( \epsilon \)-labelled outgoing edges
from all markings in the above closure is given by:

\[ \text{OutEdges}(\text{Closure}(M^{(MS,MR)}_{3(i), (0,0,0,0,0,0)})) = \{ (M^{(MS,MR)}_{4(i) \oplus MS1}, (mn', an', 0,0,0,0), \text{Send}, M^{(MS,MR)}_{3(i) \oplus MS1}, (mn', an', 0,0,0,0,0) | M^{(MS,MR)}_{4(i) \oplus MS1}, (mn', an', 0,0,0,0) \in V_f \} \]

Using the restrictions required for \( V_f \) and \( V_g \) from Equations (10.34) and (10.35) in Equation (10.37) gives:

\[ \text{OutEdges}(\text{Closure}(M^{(MS,MR)}_{3(i), (0,0,0,0,0,0)})) = \]

\[ \{ (M^{(MS,MR)}_{4(i) \oplus MS1}, (mn', an', 0,0,0,0,0), \text{Send}, M^{(MS,MR)}_{3(i) \oplus MS1}, (mn', an', 0,0,0,0,0) | mn' + an' \leq mn + an + ret' - ret - 1, 0 \leq mn' \leq mn + ret' - ret, ret' \leq ret' \leq MR \} \]

\[ \cup \{ (M^{(MS,MR)}_{1(i) \oplus MS1}, (mn', an', 0,0,0,0,0), \text{Send}, M^{(MS,MR)}_{2(i) \oplus MS1}, (mn', an', 0,0,0,0,0) | mn' + an' \leq mn + an + ret' - ret, 0 \leq mn' \leq mn + ret' - ret, ret' \leq ret' \leq MR \} \]

### 10.3 Determinisation

Now that we have expressions for the \( \epsilon \)-closures that we require, determination of the FSA interpretation of \( RG^{(MS,MR)} \), \( FSA_{RG^{(MS,MR)}} \), to obtain an equivalent deterministic FSA, \( DFSARG^{(MS,MR)} \), using lazy subset construction, proceeds as described in [72] using the formalism from Section 2.2.4.

Recall from Section 2.2.4 that a deterministic FSA, \( DFSA \), can be defined as

\[ DFSA = (S^{\text{det}}, \Sigma, \Delta^{\text{det}}, s_0^{\text{det}}, F^{\text{det}}). \]

The initial state of \( DFSA^{(MS,MR)}_{RG} \) is the \( \epsilon \)-closure of the initial marking of \( FSA_{RG^{(MS,MR)}} \), i.e. \( \text{Closure}(M^{(MS,MR)}_{1(0), (0,0,0,0,0,0)}) \). \( \text{Closure}(M^{(MS,MR)}_{1(0), (0,0,0,0,0,0)}) \), which we denote \( C_0 \), equals \( \{ M^{(MS,MR)}_{1(0), (0,0,0,0,0,0)} \} \) as there are no outgoing \( \epsilon \) 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 \text{send\_mess} transition (corresponding to the \text{Send} service primitive) is enabled in the initial marking, regardless of the value of the parameters \( MS \) and \( MR \). Thus

\[ C_0 = s_0^{\text{det}} = \text{Closure}(s_0) = \text{Closure}(M^{(MS,MR)}_{1(0), (0,0,0,0,0,0)}) = \{ M^{(MS,MR)}_{1(0), (0,0,0,0,0,0)} \} \]

Furthermore, \( C_0 \in S^{\text{det}} \).

Proceeding with lazy subset evaluation, the single state in \( C_0 \) has only one outgoing non-\( \epsilon \) arc defined. This corresponds to row 1 of Table 8.2 which maps to the \text{Send} service primitive. It leads to the state \( M^{(MS,MR)}_{(2a,0), (0,0,1,0,0,0)} \). 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.
10.3. Determinisation

The $\epsilon$-closure of $M_{(2a,0),(0,0,1,0,0)}^{(MS,MR)}$ is given by evaluating Equation (10.15) for $M_{(2a,0),(0,0,1,0,0)}^{(MS,MR)}$, which we denote $C_1$:

$$C_1 = \text{Closure}(M_{(2a,0),(0,0,1,0,0)}^{(MS,MR)}) = \{ M_{(2a,0),(0,0,1,0,0)}^{(MS,MR)} | \begin{align*} 0 \leq m'' + ao'' &\leq 0, \ 0 \leq mo'' \leq 0, \\ 0 \leq mn'' &\leq 1 + ret', 0 \leq ret' \leq MR \} \quad (10.40) $$

There are no class 3a markings because there are no old messages ($mo = 0$) and hence $mo \neq 0$. Simplifying Equation (10.40) and dropping the primes gives:

$$C_1 = \{ M_{(2a,0),(0,0,0,0,ret)}^{(MS,MR)} | 0 \leq mn \leq 1 + ret, 0 \leq ret \leq MR \} \quad (10.41)$$

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 $C_1 \in S_{det}^\theta$ and $(s_0^{det}, \text{Send}, C_1) \in \Delta_{det}^\theta$, $s_0^{det}$ and $C_1$ are illustrated in Fig. 10.5. $s_0^{det}$ 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 $\text{Send}$ 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 8.3 and Equation (10.17) the only outgoing non-epsilon edges of states in $C_1$ are edges labelled by $\text{Receive}$, corresponding to row 5 of this table. From the expression on this row the successor of a state $M_{(2a,0),(0,0,0,0,ret)}^{(MS,MR)}$ is $M_{(3b,0),(0,0,0,0,ret)}^{(MS,MR)}$ but this successor only exists if $mn > 0$, and thus the direct successors of all states in $C_1$ are captured in $V_h$:

$$V_h = \{ M_{(3b,0),(0,0,0,0,ret)}^{(MS,MR)} | M_{(2a,0),(0,0,0,0,ret)}^{(MS,MR)} \in C_1, mn > 0 \}$$

$$= \{ M_{(3b,0),(0,0,0,0,ret)}^{(MS,MR)} | 0 < mn \leq ret + 1, 0 \leq ret \leq MR \}$$

According to the procedure in Section 2.2.4 the successor of $C_1$ is the union of the $\epsilon$-closures of all markings in $V_h$, i.e. $\text{CLOSURE}(V_h)$. However, rather than calculate the $\epsilon$-closure of all states in $V_h$, let us investigate the $\epsilon$-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 $\epsilon$-closure of $M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}$ is given by evaluating Equation (10.36) for $M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}$, which we denote $C_2$:

$$C_2 = \text{Closure}(M_{(3b,0),(0,0,0,0,0)}^{(MS,MR)}) = \{ M_{(3b,0),(0,0,0,0,ret)}^{(MS,MR)} | ao \leq 0, 0 \leq mn + an \leq ret, \\ 0 \leq mn \leq ret, 0 \leq ret \leq MR \}$$

$$\cup \{ M_{(2b,0),(0,0,0,0,ret)}^{(MS,MR)} | ao \leq 0, 0 \leq mn + an \leq ret + 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR \}$$

$$\cup \{ M_{(4,3),(0,0,0,0,0)}^{(MS,MR)} | 0 \leq mn + an \leq ret - 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR \}$$

$$\cup \{ M_{(1,1),(0,0,0,0,0)}^{(MS,MR)} | 0 \leq mn + an \leq ret, 0 \leq mn \leq ret, 0 \leq ret \leq MR \} \quad (10.42)$$

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 (10.42) becomes redundant. This is because the inequality
10.3. Determinisation

\[ C_0 = \{M^{(MS,MR)}(1,0),(MR,0,0,0,0)\} \]

Figure 10.5: Construction of a deterministic FSA using subset construction with lazy evaluation, showing the initial state, \( s_0^{detr} \), and its only successor, \( C_1 \).

\[ mn'' + an'' \leq ret' - 1 \] 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 (10.42) is also redundant. Equation (10.42) 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 (10.42) gives:

$$C_2 = \{ M^{(MS,MR)}_{(3b,0),(0,0,mn,an,ret)} \mid 0 \leq mn + an \leq ret, 0 \leq ret \leq MR \}$$

$$\cup \{ M^{(MS,MR)}_{(2b,0),(0,0,mn,an,ret)} \mid 0 \leq mn + an \leq ret + 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR \}$$

$$\cup \{ M^{(MS,MR)}_{(1b),(mn,an,0,0,0)} \mid 0 \leq mn \leq MR - 1 \}$$

$$\cup \{ M^{(MS,MR)}_{(1b),(mn,an,0,0,0)} \mid 0 \leq mn + an \leq MR \}$$

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

**Lemma 10.3.** $C_2 = \text{CLOSURE}(V_h)$.

**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 8.1, for the case $ao = 0$. Given that there are no old acknowledgements in any of the states in $V_h$, and that from Equation (10.36) the number of old acknowledgements of the argument marking only affects the number of old acknowledgements in the markings of its $\epsilon$-closure ($ao'$ appears only in the inequality $ao' \leq ao$) then $C_2$ must include all possible class 3b markings from the $\epsilon$-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 8.1 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 8.1 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 $\epsilon$-closure of any marking in $V_h$.

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

**Corollary 10.3.** $C_2 \in S^{det}$ and $(C_1, \text{Receive}, C_2) \in \Delta^{det}$

This result is illustrated in Fig. 10.6. Note that although the set $C_2$ comprises states from $V^{(MS,MR)}_{(2b,0)}$, $V^{(MS,MR)}_{(3b,0)}$, $V^{(MS,MR)}_{(1,1)}$ and $V^{(MS,MR)}_{(4,1)}$, the states in all four of these subsets in Fig. 10.6 are part of the single successor state of $C_1$. Hence, in Fig. 10.6, the four shaded subsets representing $C_2$ are connected via solid lines with no arrowheads. Note that to reflect Equation (10.43), the depiction of the subsets of
Figure 10.6: Construction of a deterministic FSA showing the addition of $C_2$, the successor of $C_1$ from Fig. 10.5.
10.3. Determinisation

\[ C_2 \text{ in } V^{(MS,MR)}_{(3b,0)} \text{ and } V^{(MS,MR)}_{(2b,0)} \text{ do not cover all of the markings in } V^{(MS,MR)}_{(3b,0)} \text{ or } V^{(MS,MR)}_{(2b,0)}, \text{ whereas the subsets of } C_2 \text{ in } V^{(MS,MR)}_{(1,1)} \text{ and } V^{(MS,MR)}_{(4,1)} \text{ do.} \]

The only outgoing non-\(\epsilon\) 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 8.2 and Table 8.7. In the CPN these correspond to the \(\text{send\_mess}\) 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 \(\epsilon\) moves (and hence their destination states are also in \(C_2\)).

Because \(\text{CLOSURE}(V_h) = C_2\) then \(\text{OutEdges}(C_2) = \text{OutEdges}(\text{CLOSURE}(V_h)) = \text{OutEdges}(\text{Closure}(M^{(MS,MR)}_{(3b,0),(0,0,0,0,0)}))\). Hence, evaluating Equation (10.38) for \(M^{(MS,MR)}_{(3b,0),(0,0,0,0,0)}\) gives all outgoing non-\(\epsilon\) arcs from the markings in \(C_2\).

As an aside, we need not have chosen the marking \(M^{(MS,MR)}_{(3b,0),(0,0,0,0,0)}\) for this purpose. We may have chosen any \(M^{(MS,MR)}_{(3b,0),(0,0,\text{mn},\text{ret})} \in V_h\) provided \(\text{mn} = \text{ret}\). The reason for this can be seen in Equation (10.36). 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 \(\text{OutEdges}\) 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 \(\text{mn} = \text{ret}\) implies that no new messages have yet been lost, and we can still retransmit another \(\text{MR} - \text{ret}\) times, thus a class 3b and 2b marking with the maximum allowable value of \(\text{mn} = \text{MR}\) can be reached. Because it is only the class 1 and class 4 markings from \(C_2\) that have outgoing non-\(\epsilon\) arcs, then this guarantees that all outgoing non-\(\epsilon\) arcs from markings in \(C_2\) are obtained.

For consistency, however, we retain the choice of \(\text{Closure}(M^{(MS,MR)}_{(3b,0),(0,0,0,0,0)})\) as the argument to \(\text{OutEdges}\). The resulting arcs, given by Equation (10.38), have successor states given by \(V_j\):

\[ V_j = \{ s' \mid (s, \text{Send}, s') \in \text{OutEdges}(\text{Closure}(M^{(MS,MR)}_{(3b,0),(0,0,0,0,0)})) \} \]

\[ = \{ M^{(MS,MR)}_{(3a,0\oplus MS,1),(\text{mn},\text{an},1,0,0)} \mid 0 \leq \text{mn} + \text{an} \leq \text{ret} - 1, 0 \leq \text{mn} \leq \text{ret}, 0 \leq \text{ret} \leq \text{MR} \} \]

\[ \cup \{ M^{(MS,MR)}_{(2a,0\oplus MS,1),(\text{mn},\text{an},1,0,0)} \mid 0 \leq \text{mn} + \text{an} \leq \text{ret}, 0 \leq \text{mn} \leq \text{ret}, 0 \leq \text{ret} \leq \text{MR} \} \]

(10.44)

Performing similar simplifications to those performed in Equation (10.43), Equation (10.44) can be simplified to become:

\[ V_j = \{ M^{(MS,MR)}_{(3a,1),(\text{mn},\text{an},1,0,0)} \mid \text{mn} + \text{an} \leq \text{MR} - 1, \text{MR} > 0 \} \]

\[ \cup \{ M^{(MS,MR)}_{(2a,1),(\text{mn},\text{an},1,0,0)} \mid \text{mn} + \text{an} \leq \text{MR} \} \]

(10.45)

According to the procedure in Section 2.2.4 the successor of \(C_2\) is the union of the \(\epsilon\)-closures of all markings in \(V_j\), i.e. \(\text{CLOSURE}(V_j)\). 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. 10.6 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 (10.45):

$$V_{(k,i)} = \{ M^{(MS,MR)}_{(3a,i),(mo,ao,1,0,0)} \mid 0 \leq mo + ao \leq MR - 1 \}$$

$$\cup \{ M^{(MS,MR)}_{(2a,i),(mo,ao,1,0,0)} \mid 0 \leq mo + ao \leq MR \}$$

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 $\varepsilon$-closure of $V_{(k,i)}$, $\text{CLOSURE}(V_{(k,i)})$, we use a similar procedure to that used to discover the $\varepsilon$-closure, $C_2$. We select an appropriate single marking in $V_{(k,i)}$, calculate its $\varepsilon$-closure, and demonstrate that it covers all possible markings that could be in the $\varepsilon$-closure of any marking from $V_{(k,i)}$. The marking we have selected is the marking $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)}$ 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 $\varepsilon$-closure will span more markings than any other marking in $V_{(k,i)}$.

The $\varepsilon$-closure of $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)}$ is given by evaluating Equation (10.15) for $M^{(MS,MR)}_{(2a,i),(MR,0,1,0,0)}$, resulting in a set of markings (specified in terms of $i$) which we denote $C_{(3,i)}$:

$$C_{(3,i)} = \{ M^{(MS,MR)}_{(2a,i),(mo,ao,1,0,0,ret)} \mid 0 \leq mo + ao \leq MR, 0 \leq mo \leq MR, 0 \leq mn \leq 1 + ret,$$

$$0 \leq ret \leq MR \}$$

$$\cup \{ M^{(MS,MR)}_{(3a,i),(mo,ao,1,0,0,ret)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mo \leq MR - 1, 0 \leq mn \leq 1 + ret,$$

$$0 \leq ret \leq MR, MR > 0 \}$$

(10.47)

where the appearance of $i$ in the subscript shows that $C_{(3,i)}$ is parametric. Simplifying Equation (10.47) to remove redundant inequalities gives:

$$C_{(3,i)} = \{ M^{(MS,MR)}_{(2a,i),(mo,ao,1,0,0,ret)} \mid 0 \leq mo + ao \leq MR, 0 \leq mn \leq 1 + ret, 0 \leq ret \leq MR \}$$

$$\cup \{ M^{(MS,MR)}_{(3a,i),(mo,ao,1,0,0,ret)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mn \leq 1 + ret, 0 \leq ret \leq MR \}$$

(10.48)

where we drop the redundant $MR > 0$ term. By inspection, this $\varepsilon$-closure covers every class 2a and 3a marking defined in rows 2 and 4 of Table 8.1, for $i \in \{0, 1, ..., MS\}$, i.e.

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

(10.49)

**Lemma 10.4.** $C_{(3,i)} = \text{CLOSURE}(V_{(k,i)})$.

**Proof.** From Tables 8.3 and 8.5 and the mapping $\text{Prim}$, the destination marking of every outgoing $\varepsilon$-labelled arc with its source marking in $C_{(3,i)}$ is also in $C_{(3,i)}$, i.e. $\text{CLOSURE}(C_{(3,i)}) = C_{(3,i)}$. 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 8.3. 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 $V_{(k,i)}$, $\text{CLOSURE}(V_{(k,i)})$, will contain only class 2a and 3a markings with a sender sequence number of $i$. Because $C_{(3,i)}$ is the $\epsilon$-closure of $M_{(2a,i),(MR,0,0,1,0,0)}^{(MS,MR)} \in V_{(k,i)}$ and $C_{(3,i)}$ contains all class 2a and 3a markings with a sender sequence number of $i$, the lemma is proved. \hfill $\square$

**Corollary 10.4.** From Lemma 10.4 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. 10.7, 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 10.4, the only outgoing non-$\epsilon$ 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 8.3). Because $C_{(3,i)} = \text{Closure}(M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)})$ this means that $\text{OutEdges}(\text{Closure}(M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)}))$ will give the successors of all outgoing non-$\epsilon$ edges from states in $C_{(3,i)}$. By making the appropriate substitutions into Equation 10.17, these are given by $V_{(l,i)}$: 

$$V_{(l,i)} = \{ s' \mid (s, \text{Receive}, s') \in \text{OutEdges}(\text{Closure}(M_{(2a,i),(MR,0,1,0,0)}^{(MS,MR)})) \}$$

(10.50)

According to the procedure in Section 2.2.4, the successor of $C_{(3,i)}$ is the union of the $\epsilon$-closures of all markings in $V_{(l,i)}$. However, we again use the same approach as previously and calculate the $\epsilon$-closure of $M_{(3b,i),(0,MR,0,0,0,0)}^{(MS,MR)}$ and demonstrate that this equals $\text{CLOSURE}(V_{(l,i)})$. The marking we have chosen is $M_{(3b,i),(0,MR,0,0,0,0)}^{(MS,MR)}$, 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 $\epsilon$-closure will span more markings (successively enable more $\epsilon$ transitions) than any other marking in $V_{(l,i)}$.

The $\epsilon$-closure of $M_{(3b,i),(0,MR,0,0,0,0)}^{(MS,MR)}$ is given by evaluating Equation (10.36) for $M_{(3b,i),(0,MR,0,0,0,0)}^{(MS,MR)}$, resulting in a set of markings (specified in terms of $i$) which we denote $C_{(4,i)}$:

$$C_{(4,i)} = \text{Closure}(M_{(3b,i),(0,MR,0,0,0,0)}^{(MS,MR)})$$

$$= \{ M_{(3b,i),(0,ao,mm,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq mn \leq ret, 0 \leq ret \leq MR \}$$

$$\cup \{ M_{(2b,i),(0,ao,mm,an,ret)}^{(MS,MR)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret + 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR \}$$

$$\cup \{ M_{(4,i \in MS1),(mn,an,0,0,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret - 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR, MR > 0 \}$$

$$\cup \{ M_{(1,i \in MS1),(mn,an,0,0,0,0)}^{(MS,MR)} \mid 0 \leq mn + an \leq ret, 0 \leq mn \leq ret, 0 \leq ret \leq MR \}$$

(10.51)
$C_0 = \{ M_{(1,0),(MR,0,0,0,0)}^{(MS,MR)} \}$

Figure 10.7: Construction of a deterministic FSA showing the addition of $C_{(3,1)}$, the successor of $C_2$ from Fig. 10.6.
10.3. Determinisation

Simplifying Equation (10.51) to eliminate redundant inequalities gives:

\[
C_{(4,i)} = \{ M^{(MS,MR)}_{(3b,i),(0,ao,an,ret)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret, 0 \leq ret \leq MR \} \\
\cup \{ M^{(MS,MR)}_{(2b,i),(0,ao,an,ret)} \mid 0 \leq ao \leq MR, 0 \leq mn + an \leq ret + 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR \} \\
\cup \{ M^{(MS,MR)}_{(4,i+\oplus MS), (mn,an,0,0,0)} \mid 0 \leq mn + an \leq MR - 1, MR > 0 \} \\
\cup \{ M^{(MS,MR)}_{(1,i+\oplus MS), (mn,an,0,0,0)} \mid 0 \leq mn + an \leq MR \}
\] (10.52)

By inspection, this \(\epsilon\)-closure covers every class 2b and 3b marking (for sender sequence number = \(i\)) and every class 1 and 4 marking (for sender sequence number = \(i \oplus MS\)) defined in rows 1, 3, 5 and 6 of Table 8.1, i.e.

\[
C_{(4,i)} = V^{(MS,MR)}_{(2b,i)} \cup V^{(MS,MR)}_{(3b,i)} \cup V^{(MS,MR)}_{(1,i+MS)} \cup V^{(MS,MR)}_{(4,i+MS)}
\]

**Lemma 10.5.** \(C_{(4,i)} = Closure(V_{(i,i)})\).

**Proof.** From Tables 8.2, 8.4, 8.6 and 8.7, the destination marking of every outgoing \(\epsilon\)-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 8.2 and 8.7. They correspond to the occurrence of the **Send** transition.

Hence, because \(V_{(i,i)}\) contains only class 2b and 3b markings (with sender sequence number = \(i\)) and class 1 and 4 markings (with sender sequence number = \(i \oplus MS\)), then the closure of all markings in \(V_{(i,i)}\), \(Closure(V_{(i,i)})\), will contain only class 2b and 3b markings (with sender sequence number = \(i\)) and class 1 and 4 markings (with sender sequence number = \(i \oplus MS\)). Because \(C_{(4,i)}\) is the \(\epsilon\)-closure of \(M^{(MS,MR)}_{(3b,i),(0,MR,0,0,0)} \in V_{(i,i)}\) and it contains all class 2b, 3b, 1 and 4 markings as described, the lemma is proved.

**Corollary 10.5.** From Lemma 10.5 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. 10.8. Unlike \(C_2\), \(C_{(4,1)}\) covers all class 2b and class 3b markings in \(V^{(MS,MR)}_{1}\), whereas \(C_2\) only covers some of the class 2b and class 3b markings in \(V^{(MS,MR)}_{0}\). (However, both \(C_2\) and \(C_{(4,1)}\) cover all of the class 1 and class 4 markings, in \(V^{(MS,MR)}_{1}\) and \(V^{(MS,MR)}_{2}\) respectively.) It is important to note, for when we construct our parametric deterministic FSA, that \(C_{(4,0)}\) (obtained substituting \(i = 0\) into Equation (10.52)) and \(C_2\) are not equal.

As mentioned in the proof of Lemma 10.5, the only outgoing non-\(\epsilon\) 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 8.2 and 8.7. \(OutEdges(Closure(M^{(MS,MR)}_{(3b,i),(0,MR,0,0,0)}))\) gives exactly these edges, where in fact any marking
10.3. Determinisation

Figure 10.8: Construction of a deterministic FSA showing the addition of $C_{(4,1)}$, the successor of $C_{(3,1)}$ from Fig. 10.7.
10.3. Determinisation

\(M^{(MS,MR)}_{(3b,i),(0,MR,0,0,ret)} \in C(4,i)\) could have been used, provided \(mn = ret\), for the same reasons as previously described. The successor states of these edges, from \(OutEdges(Closure(M^{(MS,MR)}_{(3b,i),(0,MR,0,0,0)}))\) (see Equation (10.38)) are given by \(V_{(p,i)}\):

\[
V_{(p,i)} = \{ s' \mid (s, Send, s') \in OutEdges(Closure(M^{(MS,MR)}_{(3b,i),(0,MR,0,0,0)})) \}
= M^{(MS,MR)}_{(3a,i \oplus MS1),(mn,an,1,0,0)} \mid 0 \leq mn + an \leq ret - 1, 0 \leq mn \leq ret, 0 \leq ret \leq MR
\cup M^{(MS,MR)}_{(2a,i \oplus MS1),(mn,an,1,0,0)} \mid 0 \leq mn + an \leq ret, 0 \leq mn \leq ret, 0 \leq ret \leq MR \quad (10.53)
\]

Simplifying Equation (10.53) to remove redundant inequalities gives:

\[
V_{(p,i)} = M^{(MS,MR)}_{(3a,i \oplus MS1),(mn,an,1,0,0)} \mid 0 \leq mn + an \leq MR - 1, MR > 0
\cup M^{(MS,MR)}_{(2a,i \oplus MS1),(mn,an,1,0,0)} \mid 0 \leq mn + an \leq MR \quad (10.54)
\]

According to the procedure in Section 2.2.4, the successor of \(C(4,i)\) is the union of the \(\epsilon\)-closures of all markings in \(V_{(p,i)}\). However, as previously, we carefully select one marking in \(V_{(p,i)}\) and demonstrate that its \(\epsilon\)-closure is equal to \(CLOSURE(V_{(p,i)})\).

The marking we have chosen is \(M^{(MS,MR)}_{(2a,i \oplus MS1),(MR,0,1,0,0)}\). The motivation for choosing this particular marking is similar to the previous cases: we select a marking that will give us the largest \(\epsilon\)-closure out of all the markings in \(V_{(p,i)}\). The \(\epsilon\)-closure of \(M^{(MS,MR)}_{(2a,i \oplus MS1),(MR,0,1,0,0)}\) is given by evaluating Equation (10.15) for \(M^{(MS,MR)}_{(2a,i \oplus MS1),(MR,0,1,0,0)}\), resulting in a set of markings (specified in terms of \(i\)) which we denote \(C(5,i)\) (after dropping the primes):

\[
C_{(5,i)} = Closure(M^{(MS,MR)}_{(2a,i \oplus MS1),(MR,0,1,0,0)} \quad (10.55)
\]

Simplifying Equation (10.55) to eliminate redundant inequalities gives:

\[
C_{(5,i)} = M^{(MS,MR)}_{(2a,i \oplus MS1),(mo,ao,mn,0,ret)} \mid 0 \leq mo + ao \leq MR, 0 \leq mo \leq MR, 0 \leq mn \leq 1 + ret,
0 \leq ret \leq MR
\cup M^{(MS,MR)}_{(3a,i \oplus MS1),(mo,ao,mn,0,ret)} \mid 0 \leq mo + ao \leq MR - 1, 0 \leq mo \leq MR - 1, 0 \leq mn \leq ret + 1,
0 \leq ret \leq MR, MR > 0
\quad (10.56)
\]

and when replacing \(i\) with \(i \oplus MS 1\) in Table 8.1 we have

\[
C_{(5,i)} = V^{(MS,MR)}_{(2a,i \oplus MS1)} \cup V^{(MS,MR)}_{(3a,i \oplus MS1)} \quad (10.57)
\]

202
Lemma 10.6. \( C_{(5,i)} = \text{CLOSURE}(V_{(p,i)}) \).

Proof. Because \( C_{(5,i)} \) includes all possible class 2a and 3a markings defined by Table 8.1 (with sender sequence number \( i \oplus_{MS} 1 \)) and from Tables 8.3 and 8.5 and mapping \( \text{Prim} \), the only outgoing arcs with destination states not in \( V_{(MS,MR)}^{(2a,i@MS1)} \cup V_{(3a,i@MS1)}^{(MS,MR)} \) 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 \( V_{(p,i)} \), \( \text{CLOSURE}(V_{(p,i)}) \), will contain only class 2a and 3a markings with the same sender sequence number of \( i@MS1 \). Because \( C_{(5,i)} \) is the \( c \)-closure of \( M_{(MS,MR)}^{(2a,i@MS1),(MR,0,1,0,0)} \in V_{(p,i)} \) and \( C_{(5,i)} \) contains all class 2a and 3a markings with a sender sequence number of \( i@MS1 \), the lemma is proved. \( \square \)

Corollary 10.6. From Lemma 10.6 when \( i = 1 \), \( C_{(5,1)} \in S_{\text{det}} \) and hence \( (C_{(4,1)}, \text{Send}, C_{(5,1)}) \in \Delta_{\text{det}} \).

This result is illustrated in Fig. 10.9 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 (10.48) and (10.56), we find that \( C_{(5,i)} \) is equal to \( C_{(3,i@MS1)} \). We can now state and prove a lemma that builds the rest of the structure of our deterministic parametric FSA through lazy evaluation.

Lemma 10.7. \( \forall i \in \{0, 1, ..., MS\} \), \( C_{(3,i)} \in S_{\text{det}} \), \( C_{(4,i)} \in S_{\text{det}} \), \( (C_{(3,i)}, \text{Receive}, C_{(4,i)}) \in \Delta_{\text{det}} \), and \( (C_{(4,i)}, \text{Send}, C_{(3,i@MS1)}) \in \Delta_{\text{det}} \).

Proof. We know from direct construction and Corollary 10.3 that \( C_{2} \in S_{\text{det}} \) and from Corollary 10.4 that when \( i = 1 \) is substituted into Equation (10.48), we get \( C_{(3,1)} \in S_{\text{det}} \) and \( (C_{2}, \text{Send}, C_{(3,1)}) \in \Delta_{\text{det}} \).

From Corollary 10.5, we know that when substituting \( i = 1 \) into Equation (10.52), we get \( C_{(4,1)} \in S_{\text{det}} \) and \( (C_{(3,1)}, \text{Receive}, C_{(4,1)}) \in \Delta_{\text{det}} \). We know that when substituting \( i = 1 \) into Equation (10.56) and \( i = 2 \) into Equation (10.48) we get \( C_{(5,1)} = C_{(3,2)} \). Hence, from Corollary 10.6, \( C_{(3,2)} \in S_{\text{det}} \) and \( (C_{(4,1)}, \text{Send}, C_{(3,2)}) \in \Delta_{\text{det}} \).

Repeating the application of Corollaries 10.5 and 10.6 for \( i = 2, 3, ..., MS \) we get \( C_{(3,i)}, C_{(4,i)} \in S_{\text{det}} \) and \( \{(C_{(3,i)}, \text{Receive}, C_{(4,i)}), (C_{(4,i)}, \text{Send}, C_{(3,i@MS1)})\} \in \Delta_{\text{det}} \).

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, \( C_{(3,0)} \in S_{\text{det}} \). 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 \( S_{\text{det}} \).

From Lemma 10.5 and Corollary 10.5, when \( i = 0 \), we have \( C_{(4,0)} \in S_{\text{det}} \) and \( (C_{(3,0)}, \text{Receive}, C_{(4,0)}) \in \Delta_{\text{det}} \). 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 \( S_{\text{det}} \). We know that when substituting \( i = 0 \) into Equation (10.57) and \( i = 1 \) into
$C_0 = \{ M_{(1,0),(MR,0,0,0)}^{(MS,MR)} \}$

Figure 10.9: Construction of a deterministic FSA showing the addition of $C_{(5,1)}$, the successor of $C_{(4,1)}$ from Fig. 10.8.
Equation (10.49) we get $C(5,0) = C(3,1)$. Hence, from Corollary 10.6, $(C(4,0), \text{Send}, C(3,1)) \in \Delta_{\text{det}}$. Thus the lemma is proved.

Lemma 10.7, along with $C_0$ and $C_1 \in S_{\text{det}}$ and $(C_0, \text{Send}, C_1), (C_1, \text{Receive}, C_2) \in \Delta_{\text{det}}$, means we have explored all states in $S_{\text{det}}$ and all outgoing arcs of states in $S_{\text{det}}$, 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 10.2 the halt states of $FSA_{RG(MS,MR)}$ were defined as $\{M_{(1,i),(0,0,0,0)}, M_{(2a,i),(0,0,0,0,MR)} | 0 \leq i \leq MS\}$. $S_{\text{det}}$ is trivially a halt state. $C_1$ (see Equation (10.41)) is a halt state as it contains $M_{(2a,0),(0,0,0,0,MR)}$. $C_2$ (see Equation (10.43)) is a halt state as it contains $M_{(2b,0),(0,0,0,0,MR)}$. $C_3$ (Equation (10.48)) is a halt state for each $i \in \{0, 1, ..., MS\}$ because it contains $M_{(2b,i),(0,0,0,0,MR)}$. Finally, $C_{(4,i)}$ (Equation (10.52)) is a halt state for each $i \in \{0, 1, ..., MS\}$ because it contains $M_{(2b,i),(0,0,0,0,MR)}$. Thus all states of $DFSA_{RG(MS,MR)}$ are halt states.

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

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

where:

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

$DFSA_{RG(MS,MR)}$ is represented in tabular form in Table 10.1 and graphically in Fig. 10.10. By convention, the initial state is shown highlighted in bold and the halt states (all states) are shown as double circles in Fig. 10.10. 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 $C_{(3,i+MS,1)}$ on a Send event.

10.4 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 exam-
Table 10.1: $DFSA_{RG(MS,MR)}$, where rows 4 and 5 are evaluated for $0 \leq i \leq MS$.

<table>
<thead>
<tr>
<th>Source node</th>
<th>Arc Label</th>
<th>Dest. node</th>
<th>Dest. = Halt?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_0$</td>
<td>Send</td>
<td>$C_1$</td>
<td>true</td>
</tr>
<tr>
<td>$C_1$</td>
<td>Receive</td>
<td>$C_2$</td>
<td>true</td>
</tr>
<tr>
<td>$C_2$</td>
<td>Send</td>
<td>$C_{(3,1)}$</td>
<td>true</td>
</tr>
<tr>
<td>$C_{(3,i)}$</td>
<td>Receive</td>
<td>$C_{(4,i)}$</td>
<td>true</td>
</tr>
<tr>
<td>$C_{(4,i)}$</td>
<td>Send</td>
<td>$C_{(3,i@MS1)}$</td>
<td>true</td>
</tr>
</tbody>
</table>

Figure 10.10: An abstract visualisation of the parametric deterministic FSA, $DFSA_{RG(MS,MR)}$. 
10.4. Minimisation and Conformance to the SWP Service Language

Figure 10.11: An example of the deterministic FSA for $\text{MaxSeqNo}=2$. The $\text{MaxRetrans}$ parameter has no effect on the deterministic FSA.

ple in Fig. 10.11 for $MS = 2$ and arbitrary $MR$, which represents the language generated by the regular expression $(\text{Send}, \text{Receive})^* \text{Send}^1$, but which could be represented by a FSA with fewer states.

Following the minimisation procedure described in Section 2.2.4, from $DFSA_{RG(MS,MR)}$ (and Table 10.1) 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 $MFSAR_{RG(MS,MR)} = (S_{\text{min}}^{(MS,MR)}, SP, \Delta_{\text{min}}^{(MS,MR)}, 1, F_{\text{min}}^{(MS,MR)})$ where:

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

$MFSAR_{RG(MS,MR)}$ is given in tabular form in Table 10.2 and is identical to the Stop-and-Wait Service Language FSA shown in Fig. 5.3. 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
10.5 Concluding Remarks

Table 10.2: The minimised deterministic FSA, $MFSA_{RG(MS,MR)}$ representing the protocol language of $CPN_{(MS,MR)}$.

<table>
<thead>
<tr>
<th>Source node</th>
<th>Arc Label</th>
<th>Dest. node</th>
<th>Dest. = Halt?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Send</td>
<td>2</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>Receive</td>
<td>1</td>
<td>true</td>
</tr>
</tbody>
</table>

Stop-and-Wait service of $(Send, Receive)^* Send$. This verifies that the SWP does indeed satisfy the Stop-and-Wait property for all allowable values of the parameters, and thus Theorem 10.1 is proved.

10.5 Concluding Remarks

In this chapter, 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.
Chapter 11

Conclusions and Future Work

Formal verification of systems in general can be used to ensure that the system contains no logical errors and can be used to determine properties of the system. In the context of protocol engineering, this process has been formalised in a Protocol Verification Methodology [17, 23] based on a model checking approach. When considering systems with parameters, the number of states of the system tends to increase dramatically with increasing parameter values. Many techniques have been developed to alleviate state explosion, however it is often the case that the system can only be verified for small parameter values.

The parametric systems considered in this thesis are systems in which the values of one or more parameters affect the system behaviour, but the system structure remains fixed over all parameter values. When the parameters are arbitrarily large, this results in an infinite family of systems that all require verification. In this thesis, we present a new parametric approach to the verification of one such system, the Stop-and-Wait Protocol with an arbitrarily large maximum sequence number and maximum number of retransmissions.

The Stop-and-Wait Protocol is a simple flow control protocol that uses acknowledgements to both prevent buffer overflow in the receiver and as a means of transmission error recovery. It forms the basis for many more complex flow control protocols such as TCP’s Data Transfer protocol. This thesis verifies the infinite class of Stop-and-Wait protocols operating over an in-order but lossy medium. A Coloured Petri net model of the protocol parameterised with both the maximum sequence number and maximum number of retransmissions was developed and also formalised as a High-level Petri Net Graph. The properties verified parametrically (i.e. for all values of the unbounded parameters) include the absence of deadlock, absence of livelock, absence of dead transitions, and an upper bound on the channel content, as well as language analysis to verify conformance of the protocol to its service language of alternating send and receive events. Thus the infinite family of Stop-and-Wait protocols is verified to operate correctly over an in-order but lossy medium. It is possible that this approach may be developed and applied to the...
11.1 Contribution of this Research

In this section, four main areas of contribution of this thesis are summarised.

11.1.1 Algebraic Representation of an Infinite Class of Reachability Graphs

A parameterised CPN model and HLPNG specification of the Stop-and-Wait Protocol was introduced in Chapter 5. The first step toward developing an algebraic representation of the reachability graphs of the infinite family of concrete instances of this model was to analyse this model for small concrete values of the parameters. The results are shown in Chapter 6. From these results it was believed that the size of the reachability graph was factorable in the two parameters and that the size of the reachability graph grows linearly in the maximum sequence number and quartically in the maximum number of retransmissions. A conjecture on the number of nodes and number of arcs in the reachability graph was later proved to be correct (in Chapter 9). This gave insight into the regular structure of the reachability graph in both parameters. A series of behavioural properties were then identified, based on the repeating patterns of behaviour evident in the concrete reachability graphs.

A notation, suitable for representing all distinct concrete markings and arcs for every instance of the infinite family of reachability graphs of the parametric SWP model was developed in Chapter 7. This notation was then used in Chapter 8 to formally define the parametric reachability graph of the infinite family of SWP models. This representation was proved to be a correct and exact representation of the infinite family of concrete reachability graphs. Instantiation of these expressions for any allowable values of the Maximum Sequence Number and Maximum Number of Retransmissions parameters yields the exact reachability graph of the corresponding concrete system.

11.1.2 Parametric Verification of Properties directly from the Algebraic Expressions

Chapter 9 presents the verification of properties of the parametric SWP model. These properties, proved directly from the algebraic expressions, are thus proved for every concrete instance of the Stop-and-Wait Protocol.

A formula giving the size of the reachability graph in both parameters, derived from the results in Chapter 6, was proved to be correct. This was achieved by deriving expressions for the number of markings and number of arcs directly from the algebraic expressions for the parametric reachability graph. This also provided a measure of validation of the correctness of the algebraic expressions themselves.
The absence of unexpected deadlock was proved. All dead markings present in the parametric reachability graph were identified by direct inspection of the algebraic expressions, and these were found to match exactly the set of expected dead markings.

Absence of livelock was proved by verifying that all markings in the parametric reachability graph can reach a dead marking. This property is sufficient to guarantee absence of livelock and was proved by inspection of the algebraic expressions.

Absence of unexpected dead transitions was verified by identifying, for each transition, a marking enabling that transition, that is present in all concrete instances of the parametric reachability graph. The exceptions were the \texttt{timeout\_retrans} and \texttt{receive\_dup\_ack} transitions, which will be dead if the Maximum Number of Retransmissions parameter is 0. For these two transitions, a marking enabling each of these that is present in all concrete instances of the parametric reachability graph for Maximum Number of Retransmissions greater than 0 was identified.

The upper bound on each of the message and acknowledgement channels, and on the sum of the content of the message and acknowledgement channels, was also determined directly from the algebraic expressions. This was achieved by inspection of the algebraic expressions to identify those markings that maximise the number of messages and acknowledgements, both separately and together, in the channels in any given marking.

\section*{11.1.3 Parametric Language Analysis directly on the Parametric Reachability Graph}

A further property we wished to verify was conformance of the Stop-and-Wait Protocol to its service of alternating send and receive events, for all values of the parameters. To achieve this, our algebraic expressions were translated from a parametric reachability graph to a parametric finite state automaton, capturing an infinite family of automata.

Following the conventional protocol verification methodology outlined in Chapter 3, we defined a mapping from binding elements to service primitives (send or receive events) or to $\epsilon$, the empty move, and defined the halt states. Thus the mapping from reachability graphs to finite state automata was complete.

We then successfully applied standard FSA reduction techniques manually, directly to sets of symbolic markings from the algebraic expressions. Epsilon removal and determinisation were combined through the epsilon-closure technique, combined with lazy subset evaluation, after first calculating the symbolic $\epsilon$-closures of key symbolic markings. Epsilon removal and determinisation had the effect of removing the influence of the Maximum Number of Retransmissions parameter from the infinite family of FSAs. Intuitively, this makes sense as successive send and receive events may occur after any number of retransmissions, from 0 up to the maximum number of retransmissions, and retransmission events
11.1. Contribution of this Research

map to \( \epsilon \).

Standard minimisation techniques applied to the resulting infinite family of FSAs removed the influence of the Maximum Sequence Number parameter. This also makes sense intuitively, as successive send and receive events occur in the reachability graph for each sequence number from 0 up to the maximum sequence number, but when mapping to the FSA we abstract from the sequence number and have loops of indistinguishable successive send and receive events. Minimisation reduced this family to a single, concrete FSA capturing exactly the desired sequences of send and receive events, thus proving conformance of the Stop-and-Wait protocol to the Stop-and-Wait property of alternating send and receive events for all values of the parameters.

11.1.4 Toward a Parametric Verification Methodology

The results for the SWP are promising and lead us to conjecture that the following methodology, based on the procedure used in this thesis, may be more widely applicable:

1. Identify, formalise and generalise repeating patterns of behaviour:

   - Examination of the reachability graphs of a parameterised system model for small parameter values, to discover patterns of repeating behaviour and identify a regular structure of the reachability graphs.
   - Development of a suitable notation to support the expression of all states and arcs of the reachability graph of every instantiation of the parameterised system.
   - Formalisation and generalisation of the patterns of repeating behaviour, using the developed notation, to specify the states and arcs as a set of algebraic expressions in terms of the parameters.

2. State and prove the correctness of an algebraic expression for the infinite set of reachability graphs:

   - Specification of the parametric reachability graph, representing exactly the infinite family of reachability graphs that require analysis, by means of the algebraic expressions in terms of the parameters.
   - Prove the correctness of the algebraic expressions and hence the parametric reachability graph, i.e. that the expressions do represent the exact reachability graph when instantiated for each allowable combination of parameter values. This proof involves checking, symbolically, that all states are reachable from the initial state, that no additional states can be reached that have not been captured by the algebraic expressions, and that the arcs match exactly the sets defined by the algebraic expressions.
11.2 Future Work

3. Verification of properties:

- Verify properties directly from the algebraic expressions that make up the parametric reachability graph, including the size of the reachability graph in terms of the parameters, absence of unexpected deadlock, absence of livelock, absence of dead transitions, and the channel bounds (upper bound on the number of messages and acknowledgements in the channels).
- Prove the conformance of the Stop-and-Wait Protocol to the Stop-and-Wait property of alternating Send and Receive events using parametric language analysis, by interpreting the parametric reachability graph directly as a parametric Finite State Automaton (FSA), and applying standard automata reduction techniques directly to the algebraic expressions.

11.2 Future Work

11.2.1 Extending the Analysis to Reordering Channels

The next step in developing this work is to perform the same analysis, but this time for the Stop-and-Wait Protocol operating over a reordering medium. This will present a number of additional challenges. We believe that the complexity due to the in-order nature of the medium, in the form of additional restrictions on the binding elements of transitions that send and receive messages and acknowledgments based on what is at the head of the message or acknowledgement queue, will be removed. However, this same relaxation of the behaviour will introduce additional complexity due to the less restrictive nature of the events that can occur, including actions that were previously impossible (such as receiving an old duplicate message after having received one or more instances of a new message). Furthermore, when the Stop-and-Wait Protocol operates over a reordering medium, the channel content becomes unbounded, as verified in a hand-proof in [16].

The goal would be to verify the same properties (absence of deadlock, livelock, dead transitions, channel bounds) and perform the same language analysis as conducted in this thesis, to gain a greater insight into how this method can be further developed, generalised, formalised and automated.

11.2.2 The Importance of Data Independence

The notion of data independence is mentioned briefly in Chapter 5. In essence, data independence principles dictate that to verify properties of a system involving \( n \) distinct elements of data, the model of that system must capture at least \( n + 1 \) distinct elements of data. Our model of the Stop-and-Wait Protocol abstracts completely from data (or more precisely, all elements of data have been abstracted to a single element of data, and thus it has been omitted from the model). Hence, the only properties we
verify in this thesis are properties that do not involve data.

Take, for example, the properties of loss of data, duplication of data, and in-order delivery of data. These properties state that each distinct element of data must be received exactly once by the receiver, and in the same order as sent by the sender. It may seem intuitively obvious that the Stop-and-Wait Protocol, operating over an in-order medium, satisfies these properties. This is confirmed by a hand-proof in [16]. However, when dealing with reordering channels, the situation is not so obvious. Reordering channels introduce the possibility of mistakenly receiving old duplicate messages as new messages, something that cannot be detected by examining sequences of send and receive primitives alone. Indeed, a hand proof in [16] confirms the presence of protocol behaviour that can both lose data, duplicate data, and that violate in-order delivery of data, all while remaining unknown to either the sender or receiver.

In order to prove properties that concern the order of arrival of two elements of data, data independence principles require the modelling of (at least) three distinct elements of data, one each for the two distinct elements of data, and a third which is an abstraction or representation of all other elements of data. To verify such properties using language analysis requires the development of a more complicated service specification and service language, similar to that presented in [124], and a new model of the Stop-and-Wait Protocol, incorporating two distinct elements of data and a third to represent all other elements of data. We believe this to be a substantial increase in the complexity of the concrete (and thus the parametric) reachability graphs.

11.2.3 Using Equivalence Classes to Model In-Order Channels

As mentioned in Chapter 4, the content of unbounded lossy in-order channels is modelled in [3] by a restricted class of regular expressions over an alphabet of two sequence numbers (i.e. alternating bit). Thus, in a sense, all markings whose channel content can be represented by the same regular expression are considered equivalent. It may be possible to incorporate similar ideas into the modelling and analysis of our lossy in-order channels, using equivalence classes [81,82]. Actions such as message and acknowledgement loss, and message and acknowledgement reception, can occur provided there is at least one message or acknowledgement present. These actions do not depend on how many messages or acknowledgements are present, provided there is more than one. Conversely, these actions are disabled when there are no messages or acknowledgements present. The ideas from [3] may thus translate to the CPN domain through the key idea that two markings of our Stop-and-Wait Protocol model may be equivalent if the message channel content could be represented by the same regular expression, and similarly with the acknowledgement channel content. This idea requires significant theoretical development, firstly to demonstrate that this kind of equivalence is valid for our model, then to generalise it for arbitrary systems operating over lossy in-order channels, and finally to see how this idea could be incorporated into our
algebraic expressions.

11.2.4 Developing and Automating a Parametric Verification Methodology

The Protocol Verification Methodology [17, 23], briefly described in Chapter 3, has previously been used for concrete systems. This thesis attempts to adapt the Protocol Verification Methodology for use with a specific parametric system, by developing algebraic expressions representing the infinite class of reachability graphs of that system, and then performing the analysis and verification activities directly on the algebraic expressions rather than on a single concrete reachability graph. Our attempts to adapt the Protocol Verification Methodology and the experience and insights gained in doing so may give rise to a general Parametric Verification Methodology, not exclusively for protocols, but for any amenable system. Ultimately, our goal is the development of automated tools to assist in or fully automate the steps of such a Parametric Verification Methodology.
References


REFERENCES


REFERENCES


[37] CPN ML: An Extension of Standard ML.


http://www.lsv.ens-cachan.fr/fast/.


REFERENCES


[77] ITU-T. Recommendation X.25, Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public networks by dedicated circuit. International Telecommunications Union, October 1996.
REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


Appendix A

Concepts from Algebraic Specification

The following material is based on Annex A of [73] and introduces the necessary concepts from algebraic specification that are required to define the HLPNG, including signatures, variables, terms and many-sorted algebras. This material is included for self-containment of this thesis.

A.1 Signatures

A many-sorted signature, $Sig$, is a pair:

$$Sig = (Sorts, Ops)$$

where

- $Sorts$ is a set of sorts (the names of sets, e.g. $Int$ for the integers); and
- $Ops$ is a set of operators (the names of functions) together with their arity in $Sorts$ which specifies the names of the domain and co-domain of each of the operators.
- $Sorts$ and $Ops$ are disjoint.

Arity is a function from the set of operator names to $Sorts^* \times Sorts$, where $Sorts^*$ is the set of finite sequences, including the empty string, $\epsilon$, over $Sort$. All operators in $Ops$ are denoted $o_{(\sigma,s)}$, where $\sigma \in Sorts^*$ specifies the input or argument sorts, and $s \in Sorts$ the output or range sort of operator $o$.

Using a standard convention, the sort of a constant may be declared by letting $\sigma = \epsilon$ and is denoted by $o_{(\epsilon,s)}$ or simply $o_s$.

Using the examples from [73], if $Sorts = \{Int, Bool\}$, then $o_{(Int,Int,Bool)}$ represents a binary predicate symbol, such as equality ($=$) or less than ($<$). An integer constant is denoted by $o_{(\epsilon,Int)}$ or simply $o_{Int}$. 

230
A.2 Boolean Signature

The term *Boolean Signature* refers to a many-sorted signature where:

- one of the sorts is $\text{Bool}$, i.e. $\text{Bool} \in \text{Sorts}$, which corresponds to the standard boolean set ($\text{Boolean} = \{\text{true}, \text{false}\}$) in any associated algebra; and
- one of the operators is the constant, $\text{true}_{\text{Bool}}$, i.e. $\text{true}_{\text{Bool}} \in \text{Ops}$, which corresponds to the value, true, in any associated algebra.

A.3 Variables

Let $\text{Vars}$ be a set of variables, sorted with respect to the set of sorts, $\text{Sorts}$, where:

- the sort of a variable may be declared in the same way as that of constants, i.e. a variable $v \in \text{Vars}$ of sort $s \in \text{Sorts}$ is denoted by $v_{(e,s)}$ or more simply $v_s$. Using the example from [73], if $\text{Int} \in \text{Sorts}$, then an integer variable would be $v_{(e,\text{Int})}$ or simply $v_{\text{Int}}$; and
- $\text{Vars}$ may be partitioned according to sorts, where $\text{Vars}_s$ denotes the set of variables of sort $s$, i.e. $\forall s \in \text{Sorts}, \text{Vars}_s = \{v_a \mid v_a \in \text{Vars}, a = s\}$.

A.4 Terms built from a Signature and Variables

Let $\text{TERM}(\text{Ops} \cup \text{Vars})_s$ denote the set of terms of sort $s \in \text{Sorts}$, built inductively from a signature, $\text{Sig} = (\text{Sorts}, \text{Ops})$, and a many-sorted set of variables with respect to $\text{Sorts}$, $\text{Vars}$, in the following way. For all $s \in \text{Sorts}$:

1. for all $o_{(e,s)} \in \text{Ops}, o_{(e,s)} \in \text{TERM}(\text{Ops} \cup \text{Vars})_s$;

2. $\text{Vars}_s \subseteq \text{TERM}(\text{Ops} \cup \text{Vars})_s$; and

3. for:

   (a) $s_1, \ldots, s_n \in \text{Sorts} \ (n > 0)$,
   (b) terms $e_1 \in \text{TERM}(\text{Ops} \cup \text{Vars})_{s_1}, \ldots, e_n \in \text{TERM}(\text{Ops} \cup \text{Vars})_{s_n}$ and
   (c) an operator $o_{(s_1,\ldots,s_n,s)} \in \text{Ops}$,

   $o_{(s_1,\ldots,s_n,s)}(e_1,\ldots,e_n) \in \text{TERM}(\text{Ops} \cup \text{Vars})_s$
A.5. Many-sorted Algebras

Using the example from [73], if \( \text{Int} \) is a sort, then integer constants, integer variables, and operators of output sort \( \text{Int} \) are all terms of sort \( \text{Int} \).

The set of all terms is defined as

\[
\text{TERM}(\text{Ops} \cup \text{Vars}) = \bigcup_{s \in \text{Sorts}} \text{TERM}(\text{Ops}, \text{Vars})_s
\]

A.5 Many-sorted Algebras

A many-sorted algebra (or \( \text{Sig} \)-Algebra), \( \text{Alg} \), provides an interpretation (meaning) for the signature, \( \text{Sig} \):

- For every sort, \( s \in \text{Sorts} \), there is a corresponding set, \( \text{Type}_s \), known as a carrier.

- For every operator, \( o_{(s_1, \ldots, s_n, s)} \in \text{Ops} \) \( (n > 0) \), there is a corresponding function

\[
o_{\text{Alg}} : \text{Type}_{s_1} \times \ldots \times \text{Type}_{s_n} \rightarrow \text{Type}_s
\]

- For every constant operator, \( o_s \), there is a corresponding element, \( o_{\text{Alg}} \in \text{Type}_s \), which may be considered as a function of arity zero.

**Definition A.1** (Many-Sorted Algebra).

A many-sorted Algebra, \( \text{Alg} \), for a signature, \( \text{Sig} = (\text{Sorts}, \text{Ops}) \), is a pair

\[
\text{Alg} = (\text{Types}, \text{Funs})
\]

where

- \( \text{Types} = \{\text{Type}_s \mid s \in \text{Sorts}\} \) is the set of types, called carriers, corresponding to the set of sorts in \( \text{Sig} \), with for all \( s \in \text{Sorts}, \text{Type}_s \neq \emptyset \); and

- \( \text{Funs} = \{o_{\text{Alg}} \mid o_{(\sigma, s)} \in \text{Ops}, \sigma \in \text{Sorts}^*, s \in \text{Sorts}\} \) is the set of functions corresponding to the operators in \( \text{Sig} \).

Taking the example from [73], if \( \text{Sig} = (\{\text{Int}, \text{Bool}\}, \{\langle \text{Int,Int,Bool} \rangle\}) \) then a corresponding many-sorted algebra would be

\[
\text{Alg} = (\{\text{Z}, \text{Boolean}\}, \{\text{lessthan}\})
\]

where \( \text{Z} \) is the set of integers: \( \{-\ldots, -1, 0, 1, \ldots\} \) and \( \text{Boolean} = \{\text{true, false}\} \), and \( \text{lessthan} : \text{Z} \times \text{Z} \rightarrow \text{Boolean} \) is the usual integer comparison function.

Note that sorted variables are typed by the carrier corresponding to their sort.
A.6 Assignment and Evaluation

Given a many-sorted algebra, $Alg$, and many-sorted variables (with respect to $Sorts$) in $Vars$, an assignment (sometimes called a binding or valuation) for $Alg$ and $Vars$ is a family of functions, $\alpha$, comprising an assignment function, for each sort $s \in Sorts$:

$$\alpha_s : Vars_s \rightarrow Type_s$$

This function may be extended to terms by defining the family of functions, $Val_{\alpha}$, comprising, for each sort $s \in Sorts$:

$$Val_{s,\alpha} : TERM(Ops \cup Vars)_s \rightarrow Type_s$$

The values are determined inductively as follows. For $s \in Sorts$:

1. For a constant, $o_s \in Ops$, $Val_{s,\alpha}(o_s) = o_{Alg} \in Type_s$;

2. For a variable, $v_s \in Vars_s$, $Val_{s,\alpha}(v_s) = \alpha_s(v_s)$; and

3. For $\sigma \in Sorts^* \setminus \{\epsilon\}$, $\sigma = s_1s_2...s_n$, with $s_1, ..., s_n \in Sorts$, $e \in TERM(Ops \cup Vars)$, and $e_1 \in TERM(Ops \cup Vars)_{s_1}$, ..., $e_n \in TERM(Ops \cup Vars)_{s_n}$, if $e = o_{(\sigma,s)}(e_1, ..., e_n)$, then

$$Val_{s,\alpha}(e) = o_{Alg}(Val_{s_1,\alpha}(e_1), ..., Val_{s_n,\alpha}(e_n)) \in Type_s$$
Appendix B

Reachability Graphs for \textbf{MaxRetrans=0}

B.1 Reachability Graphs

This appendix presents four reachability graphs of four concrete instances of our parametric Stop-and-Wait Protocol, including node and arc descriptors, as generated by Design/CPN. The concrete instances for which reachability graphs have been generated are for \(\text{MaxSeqNo}=1, 2, 3\) and \(4\), with \(\text{MaxRetrans}=0\) in all four cases. The node descriptors give details of the marking of each place in the CPN model, i.e. the tokens on each place, for each state (node of the graph). The arc descriptors give details of the binding element that occurred, corresponding to each arc in the reachability graph. As highlighted in Chapter 6, the repeated pattern of behaviour identified for \(\text{MaxSeqNo}=1\) and \(2\) continues for \(\text{MaxSeqNo}=3\) and \(4\).
B.1. Reachability Graphs

Figure B.1: The reachability graph, $RG(1,0)$, when $MaxSeqNo=1$ and $MaxRetrans=0$. 
Figure B.2: The reachability graph, $RG_{(2,0)}$, when MaxSeqNo=2 and MaxRetrans=0.
B.1. Reachability Graphs

Figure B.3: The reachability graph, $RG_{(3,0)}$, when MaxSeqNo=3 and MaxRetrans=0.
Figure B.4: The reachability graph, $RG_{(4,0)}$, when MaxSeqNo=4 and MaxRetrans=0.
B.2 State Space Reports

This section contains the standard state space reports generated by Design/CPN [39] for the four reachability graphs shown in Section B.1. Each state space report is divided into a number of sections. The first section provides some statistical information, giving the size of the reachability graph and strongly connected component graph, and how long it took to generate each. The second section gives the boundedness properties. These include the upper and lower integer bounds for each place, i.e. the maximum and minimum number of tokens on each place over all reachable markings, and the upper and lower multiset bounds for each place. The upper multiset bound for a place is the smallest multiset that is larger than all multisets of tokens residing on that place over all reachable markings, i.e. the union of all reachable markings of that place. The lower multiset bound is the largest multiset that is smaller than all multisets of tokens residing on that place over all reachable markings, i.e. the intersection of all reachable markings of that place. The third section gives the home properties, specifically, a list of home markings. Home markings are markings that are reachable from all other markings. Next come the liveness properties, which include a list of the dead markings, dead transition instances, and live transition instances (transitions that can always become enabled). Finally, the report lists the fairness properties, giving details of how frequently different transitions occur.

B.2.1 MaxSeqNo = 1 and MaxRetrans = 0

Statistics

<table>
<thead>
<tr>
<th>Occurrence Graph</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes:</td>
<td>12</td>
</tr>
<tr>
<td>Arcs:</td>
<td>12</td>
</tr>
<tr>
<td>Secs:</td>
<td>0</td>
</tr>
<tr>
<td>Status:</td>
<td>Full</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scc Graph</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes:</td>
<td>5</td>
</tr>
<tr>
<td>Arcs:</td>
<td>4</td>
</tr>
<tr>
<td>Secs:</td>
<td>0</td>
</tr>
</tbody>
</table>

Boundedness Properties

<table>
<thead>
<tr>
<th>Best Integers Bounds</th>
<th>Upper</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>New’ack_channel 1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
B.2. State Space Reports

New’mess_channel 1 1 1
New’receiver_state 1 1 1
New’recv_seq_no 1 1 1
New’retrans_counter 1 1 1
New’send_seq_no 1 1 1
New’sender_state 1 1 1

Best Upper Multi-set Bounds
New’ack_channel 1 1’[] ++ 1’[0] ++ 1’[1]
New’mess_channel 1 1’[] ++ 1’[0] ++ 1’[1]
New’receiver_state 1
1’r_ready ++ 1’process
New’recv_seq_no 1 1’0 ++ 1’1
New’retrans_counter 1
1’0
New’send_seq_no 1 1’0 ++ 1’1
New’sender_state 1 1’s_ready ++ 1’wait_ack

Best Lower Multi-set Bounds
New’ack_channel 1 empty
New’mess_channel 1 empty
New’receiver_state 1
empty
New’recv_seq_no 1 empty
New’retrans_counter 1
1’0
New’send_seq_no 1 empty
New’sender_state 1 empty

Home Properties

Home Markings: None

Liveness Properties

Dead Markings: [4, 7, 10, 12]
Dead Transitions Instances:
New’receive_dup_ack 1
New’timeout_retrans 1
Live Transitions Instances: None

Fairness Properties

<table>
<thead>
<tr>
<th>Event</th>
<th>Fairness</th>
</tr>
</thead>
<tbody>
<tr>
<td>New’ack_loss</td>
<td>Just</td>
</tr>
<tr>
<td>New’mess_loss</td>
<td>Just</td>
</tr>
<tr>
<td>New’receive_ack</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’receive_dup_ack</td>
<td>Fair</td>
</tr>
<tr>
<td>New’receive_mess</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’send_ack</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’send_mess</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’timeout_retrans</td>
<td>Fair</td>
</tr>
</tbody>
</table>

B.2.2 MaxSeqNo = 2 and MaxRetrans = 0

Statistics

Occurrence Graph
- Nodes: 18
- Arcs: 18
- Secs: 0
- Status: Full

Scc Graph
- Nodes: 7
- Arcs: 6
- Secs: 0

Boundedness Properties

<table>
<thead>
<tr>
<th>Event</th>
<th>Upper</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>New’ack_channel</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New’mess_channel</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New’receiver_state</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
B.2. State Space Reports

New’recv_seq_no 1 1 1
New’retrans_counter 1 1 1
New’send_seq_no 1 1 1
New’sender_state 1 1 1

Best Upper Multi-set Bounds
New’ack_channel 1 []++ 1'[0]++ 1'[1]++ 1'[2]
New’mess_channel 1 []++ 1'[0]++ 1'[1]++ 1'[2]
New’receiver_state 1
\r\n1'r_ready++ 1'process
New’recv_seq_no 1 1'0++ 1'1++ 1'2
New’retrans_counter 1
1'0
New’send_seq_no 1 1'0++ 1'1++ 1'2
New’sender_state 1 1's_ready++ 1'wait_ack

Best Lower Multi-set Bounds
New’ack_channel 1 empty
New’mess_channel 1 empty
New’receiver_state 1
empty
New’recv_seq_no 1 empty
New’retrans_counter 1
1'0
New’send_seq_no 1 empty
New’sender_state 1 empty

Home Properties
------------------------------------------------------------------------
Home Markings: None
------------------------------------------------------------------------

Liveness Properties
------------------------------------------------------------------------
Dead Markings: 6 [7,4,18,16,13,...]
Dead Transitions Instances:
New’receive_dup_ack 1

242
B.2. State Space Reports

New ‘timeout_retrans’ 1
Live Transitions Instances: None

Fairness Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Fairness</th>
</tr>
</thead>
<tbody>
<tr>
<td>New ’ack_loss’</td>
<td>Just</td>
</tr>
<tr>
<td>New ’mess_loss’</td>
<td>Just</td>
</tr>
<tr>
<td>New ’receive_ack’</td>
<td>Impartial</td>
</tr>
<tr>
<td>New ’receive_dup_ack’</td>
<td>Fair</td>
</tr>
<tr>
<td>New ’receive_mess’</td>
<td>Impartial</td>
</tr>
<tr>
<td>New ’send_ack’</td>
<td>Impartial</td>
</tr>
<tr>
<td>New ’send_mess’</td>
<td>Impartial</td>
</tr>
<tr>
<td>New ’timeout_retrans’</td>
<td>Fair</td>
</tr>
</tbody>
</table>

B.2.3 MaxSeqNo = 3 and MaxRetrans = 0

Statistics

Occurrence Graph
- Nodes: 24
- Arcs: 24
- Secs: 0
- Status: Full

Scc Graph
- Nodes: 9
- Arcs: 8
- Secs: 0

Boundedness Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Upper</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>New ’ack_channel’</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New ’mess_channel’</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New ’receiver_state’</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New ’recv_seq_no’</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New ’retrans_counter’</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
B.2. State Space Reports

New’send_seq_no 1 1 1
New’sender_state 1 1 1

Best Upper Multi-set Bounds
New’receiver_state 1
1′r_ready++ 1′process
New’recv_seq_no 1 1′0++ 1′1++ 1′2++ 1′3
New’retrans_counter 1
1′0
New’send_seq_no 1 1′0++ 1′1++ 1′2++ 1′3
New’sender_state 1 1′s_ready++ 1′wait_ack

Best Lower Multi-set Bounds
New’ack_channel 1 empty
New’mess_channel 1 empty
New’receiver_state 1
empty
New’recv_seq_no 1 empty
New’retrans_counter 1
1′0
New’send_seq_no 1 empty
New’sender_state 1 empty

Home Properties
-----------------------------------------------
Home Markings: None

Liveness Properties
-----------------------------------------------
Dead Markings: 8 [7,4,24,22,19,...]
Dead Transitions Instances:

New’receive_dup_ack 1
New’timeout_retrans 1
Live Transitions Instances: None
B.2. State Space Reports

Fairness Properties

<table>
<thead>
<tr>
<th>Event</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>New’ack_loss 1</td>
<td>Just</td>
</tr>
<tr>
<td>New’mess_loss 1</td>
<td>Just</td>
</tr>
<tr>
<td>New’receive_ack 1</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’receive_dup_ack 1</td>
<td>Fair</td>
</tr>
<tr>
<td>New’receive_mess 1</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’send_ack 1</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’send_mess 1</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’timeout_retrans 1</td>
<td>Fair</td>
</tr>
</tbody>
</table>

B.2.4 MaxSeqNo = 4 and MaxRetrans = 0

Statistics

Occurrence Graph
- Nodes: 30
- Arcs: 30
- Secs: 0
- Status: Full

Scc Graph
- Nodes: 11
- Arcs: 10
- Secs: 0

Boundedness Properties

<table>
<thead>
<tr>
<th>Event</th>
<th>Upper</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>New’ack_channel 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New’mess_channel 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New’receiver_state 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New’recv_seq_no 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New’retrans_counter 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New’send_seq_no 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New’sender_state 1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Best Upper Multi-set Bounds
New’ack_channel 1 1[…]++ 1’[0]++ 1’[1]++ 1’[2]++ 1’[3]++ 1’[4]
New’mess_channel 1 1[…]++ 1’[0]++ 1’[1]++ 1’[2]++ 1’[3]++ 1’[4]
New’receiver_state 1
   1’r_ready++ 1’process
New’recv_seq_no 1 1’0++ 1’1++ 1’2++ 1’3++ 1’4
New’retrans_counter 1
   1’0
New’send_seq_no 1 1’0++ 1’1++ 1’2++ 1’3++ 1’4
New’sender_state 1 1’s_ready++ 1’wait_ack

Best Lower Multi-set Bounds
New’ack_channel 1 empty
New’mess_channel 1 empty
New’receiver_state 1
   empty
New’recv_seq_no 1 empty
New’retrans_counter 1
   1’0
New’send_seq_no 1 empty
New’sender_state 1 empty

Home Properties
------------------------------------------------------------------------
Home Markings: None

Liveness Properties
------------------------------------------------------------------------
Dead Markings: 10 [7,4,30,28,25,...]
Dead Transitions Instances:

New’receive_dup_ack 1
New’timeout_retrans 1
Live Transitions Instances: None
B.2. State Space Reports

**Fairness Properties**

<table>
<thead>
<tr>
<th>Event Description</th>
<th>Fairness</th>
</tr>
</thead>
<tbody>
<tr>
<td>New’ack_loss 1</td>
<td>Just</td>
</tr>
<tr>
<td>New’mess_loss 1</td>
<td>Just</td>
</tr>
<tr>
<td>New’receive_ack 1</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’receive_dup_ack 1</td>
<td>Fair</td>
</tr>
<tr>
<td>New’receive_mess 1</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’send_ack 1</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’send_mess 1</td>
<td>Impartial</td>
</tr>
<tr>
<td>New’timeout_retrans 1</td>
<td>Fair</td>
</tr>
</tbody>
</table>
Appendix C

Reachability Graph Statistics

This appendix presents a table of the number of nodes and arcs in the concrete reachability graphs of our parametric Stop-and-Wait Protocol CPN model. Following this table, a derivation of a formula for the size of the state space, based on fitting polynomials to data, is presented.

C.1 Concrete Reachability Graph Statistics

This table presents the results of generating the concrete reachability graphs of our SWP CPN model instantiated for \( \text{MaxRetrans} \) from 0 to 10 (column 1) and \( \text{MaxSeqNo} \) from 1 to 11 (column 2). The number of nodes, arcs and dead markings are shown in columns 3, 4 and 5 respectively. It also shows the channel bounds (column 6) and the time taken (in seconds) for Design/CPN to generate the reachability graphs (column 7). The channel bounds are represented as pairs of integers. The first integer is the maximum number of messages that are in the message channel in any marking, and the second integer is the maximum number of acknowledgements that are in the acknowledgement channel in any marking (both obtained from the standard state space report for each scenario). The empirical evidence here, suggesting that the maximum channel bound is \( 2\text{MaxRetrans}+1 \) for both channels, is proved correct in Chapter 9. Due to the extensive nature of this table, state space reports have not been included for the scenarios analysed. All experiments were conducted on a Pentium IV 2.6GHz PC running Fedora Core 4.
C.1. Concrete Reachability Graph Statistics

Table C.1: RG statistics of the CPN in Figs. 5.5 and 5.6.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>MaxSeqNo</th>
<th>nodes</th>
<th>arcs</th>
<th>Dead Nodes</th>
<th>Channel bounds</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>12</td>
<td>12</td>
<td>4</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>18</td>
<td>18</td>
<td>6</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>24</td>
<td>24</td>
<td>8</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>30</td>
<td>30</td>
<td>10</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>36</td>
<td>36</td>
<td>12</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>42</td>
<td>42</td>
<td>14</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>48</td>
<td>48</td>
<td>16</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>54</td>
<td>54</td>
<td>18</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>60</td>
<td>60</td>
<td>20</td>
<td>(1,1)</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>66</td>
<td>66</td>
<td>22</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>72</td>
<td>72</td>
<td>24</td>
<td>(1,1)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>92</td>
<td>242</td>
<td>4</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>138</td>
<td>363</td>
<td>6</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>184</td>
<td>484</td>
<td>8</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>230</td>
<td>605</td>
<td>10</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>276</td>
<td>726</td>
<td>12</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>322</td>
<td>847</td>
<td>14</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>368</td>
<td>968</td>
<td>16</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>414</td>
<td>1089</td>
<td>18</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>460</td>
<td>1210</td>
<td>20</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>506</td>
<td>1331</td>
<td>22</td>
<td>(3,3)</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>552</td>
<td>1452</td>
<td>24</td>
<td>(3,3)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>336</td>
<td>1166</td>
<td>4</td>
<td>(5,5)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>504</td>
<td>1749</td>
<td>6</td>
<td>(5,5)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>672</td>
<td>2332</td>
<td>8</td>
<td>(5,5)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>840</td>
<td>2915</td>
<td>10</td>
<td>(5,5)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1008</td>
<td>3498</td>
<td>12</td>
<td>(5,5)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1176</td>
<td>4081</td>
<td>14</td>
<td>(5,5)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>1344</td>
<td>4664</td>
<td>16</td>
<td>(5,5)</td>
<td>0</td>
</tr>
</tbody>
</table>

continued on next page
| MaxRetrans | MaxSeqNo | |nodes| |arcs| | Dead Nodes | Channel bounds | Time (s) |
|-----------|---------|---------|------|------|------------|-------------|---------|
| 2         | 8       | 1512    | 5247 | 18   | (5,5)      | 0           |         |
| 2         | 9       | 1680    | 5830 | 20   | (5,5)      | 1           |         |
| 2         | 10      | 1848    | 6413 | 22   | (5,5)      | 1           |         |
| 2         | 11      | 2016    | 6996 | 24   | (5,5)      | 0           |         |
| 3         | 1       | 880     | 3494 | 4    | (7,7)      | 1           |         |
| 3         | 2       | 1320    | 5241 | 6    | (7,7)      | 0           |         |
| 3         | 3       | 1760    | 6988 | 8    | (7,7)      | 1           |         |
| 3         | 4       | 2200    | 8735 | 10   | (7,7)      | 1           |         |
| 3         | 5       | 2640    | 10482| 12   | (7,7)      | 1           |         |
| 3         | 6       | 3080    | 12229| 14   | (7,7)      | 1           |         |
| 3         | 7       | 3520    | 13976| 16   | (7,7)      | 2           |         |
| 3         | 8       | 3960    | 15723| 18   | (7,7)      | 2           |         |
| 3         | 9       | 4400    | 17470| 20   | (7,7)      | 2           |         |
| 3         | 10      | 4840    | 19217| 22   | (7,7)      | 2           |         |
| 3         | 11      | 5280    | 20964| 24   | (7,7)      | 3           |         |
| 4         | 1       | 1900    | 8176 | 4    | (9,9)      | 1           |         |
| 4         | 2       | 2850    | 12264| 6    | (9,9)      | 2           |         |
| 4         | 3       | 3800    | 16352| 8    | (9,9)      | 2           |         |
| 4         | 4       | 4750    | 20440| 10   | (9,9)      | 3           |         |
| 4         | 5       | 5700    | 24528| 12   | (9,9)      | 4           |         |
| 4         | 6       | 6650    | 28616| 14   | (9,9)      | 5           |         |
| 4         | 7       | 7600    | 32704| 16   | (9,9)      | 6           |         |
| 4         | 8       | 8550    | 36792| 18   | (9,9)      | 7           |         |
| 4         | 9       | 9500    | 40880| 20   | (9,9)      | 9           |         |
| 4         | 10      | 10450   | 44968| 22   | (9,9)      | 10          |         |
| 4         | 11      | 11400   | 49056| 24   | (9,9)      | 12          |         |
| 5         | 1       | 3612    | 16402| 4    | (11,11)    | 2           |         |
| 5         | 2       | 5418    | 24603| 6    | (11,11)    | 4           |         |
| 5         | 3       | 7224    | 32804| 8    | (11,11)    | 7           |         |

continued on next page
### Table C.1 – continued from previous page

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>MaxSeqNo</th>
<th>nodes</th>
<th>arcs</th>
<th>Dead Nodes</th>
<th>Channel bounds</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4</td>
<td>9030</td>
<td>41005</td>
<td>10</td>
<td>(11,11)</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10836</td>
<td>49206</td>
<td>12</td>
<td>(11,11)</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>12642</td>
<td>57407</td>
<td>14</td>
<td>(11,11)</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>14448</td>
<td>65608</td>
<td>16</td>
<td>(11,11)</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>16254</td>
<td>73809</td>
<td>18</td>
<td>(11,11)</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>18060</td>
<td>82010</td>
<td>20</td>
<td>(11,11)</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>19866</td>
<td>90211</td>
<td>22</td>
<td>(11,11)</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>21672</td>
<td>98412</td>
<td>24</td>
<td>(11,11)</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6272</td>
<td>29602</td>
<td>4</td>
<td>(13,13)</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>9408</td>
<td>44403</td>
<td>6</td>
<td>(13,13)</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>12544</td>
<td>59204</td>
<td>8</td>
<td>(13,13)</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>15680</td>
<td>74005</td>
<td>10</td>
<td>(13,13)</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>18816</td>
<td>88806</td>
<td>12</td>
<td>(13,13)</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>21952</td>
<td>103607</td>
<td>14</td>
<td>(13,13)</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>25088</td>
<td>118408</td>
<td>16</td>
<td>(13,13)</td>
<td>50</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>28224</td>
<td>133209</td>
<td>18</td>
<td>(13,13)</td>
<td>60</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>31360</td>
<td>148010</td>
<td>20</td>
<td>(13,13)</td>
<td>72</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>34496</td>
<td>162811</td>
<td>22</td>
<td>(13,13)</td>
<td>89</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>37632</td>
<td>177612</td>
<td>24</td>
<td>(13,13)</td>
<td>99</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>10176</td>
<td>49446</td>
<td>4</td>
<td>(15,15)</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>15264</td>
<td>74169</td>
<td>6</td>
<td>(15,15)</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>20352</td>
<td>98892</td>
<td>8</td>
<td>(15,15)</td>
<td>38</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>25440</td>
<td>123615</td>
<td>10</td>
<td>(15,15)</td>
<td>56</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>30528</td>
<td>148338</td>
<td>12</td>
<td>(15,15)</td>
<td>77</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>35616</td>
<td>173061</td>
<td>14</td>
<td>(15,15)</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>40704</td>
<td>197784</td>
<td>16</td>
<td>(15,15)</td>
<td>125</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>45792</td>
<td>222507</td>
<td>18</td>
<td>(15,15)</td>
<td>150</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>50880</td>
<td>247230</td>
<td>20</td>
<td>(15,15)</td>
<td>181</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>55968</td>
<td>271953</td>
<td>22</td>
<td>(15,15)</td>
<td>220</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>61056</td>
<td>296676</td>
<td>24</td>
<td>(15,15)</td>
<td>249</td>
</tr>
</tbody>
</table>

*continued on next page*
| MaxRetrans | MaxSeqNo | |nodes| |arcs| | Dead Nodes | Channel bounds | Time (s) |
|-----------|----------|------------------|----------|----------|----------|-----------------|-----------------|----------|
| 8         | 1        | 15660            | 77844    | 4        | (17,17)  | 25               |
| 8         | 2        | 23490            | 116766   | 6        | (17,17)  | 51               |
| 8         | 3        | 31320            | 155688   | 8        | (17,17)  | 84               |
| 8         | 4        | 39150            | 194610   | 10       | (17,17)  | 125              |
| 8         | 5        | 46980            | 233532   | 12       | (17,17)  | 178              |
| 8         | 6        | 54810            | 272454   | 14       | (17,17)  | 223              |
| 8         | 7        | 62640            | 311376   | 16       | (17,17)  | 275              |
| 8         | 8        | 70470            | 350298   | 18       | (17,17)  | 350              |
| 8         | 9        | 78300            | 389220   | 20       | (17,17)  | 409              |
| 8         | 10       | 86130            | 428142   | 22       | (17,17)  | 497              |
| 8         | 11       | 93960            | 467064   | 24       | (17,17)  | 557              |
| 9         | 1        | 23100            | 116946   | 4        | (19,19)  | 48               |
| 9         | 2        | 34650            | 175419   | 6        | (19,19)  | 102              |
| 9         | 3        | 46200            | 233892   | 8        | (19,19)  | 169              |
| 9         | 4        | 57750            | 292365   | 10       | (19,19)  | 258              |
| 9         | 5        | 69300            | 350838   | 12       | (19,19)  | 357              |
| 9         | 6        | 80850            | 409311   | 14       | (19,19)  | 493              |
| 9         | 7        | 92400            | 467784   | 16       | (19,19)  | 632              |
| 9         | 8        | 103950           | 526257   | 18       | (19,19)  | 723              |
| 9         | 9        | 115500           | 584730   | 20       | (19,19)  | 857              |
| 9         | 10       | 127050           | 643203   | 22       | (19,19)  | 1010             |
| 9         | 11       | 138600           | 701676   | 24       | (19,19)  | 1172             |
| 10        | 1        | 32912            | 169142   | 4        | (21,21)  | 92               |
| 10        | 2        | 49368            | 253713   | 6        | (21,21)  | 194              |
| 10        | 3        | 65824            | 338284   | 8        | (21,21)  | 328              |
| 10        | 4        | 82280            | 422855   | 10       | (21,21)  | 494              |
| 10        | 5        | 98736            | 507426   | 12       | (21,21)  | 681              |
| 10        | 6        | 115192           | 591997   | 14       | (21,21)  | 978              |
| 10        | 7        | 131648           | 676568   | 16       | (21,21)  | 1205             |

continued on next page
### C.2 The Size of the Reachability Graph

By making the assumption that the size of the reachability graph is polynomial and factorable in the two parameters, we can use a method for fitting polynomials to data, based on successive differences, to find this polynomial formula. (We can apply this method in any case, but it will fail if the size of the RG is not polynomial.)

#### C.2.1 The Number of Nodes

We shall first attempt to fit a polynomial to our $|V_{(MS,MR)}|$ data in the MaxRetrans parameter, for fixed MaxSeqNo.

The case where $\text{MaxSeqNo} = 1$

Table C.2 contains the successive difference in the number of nodes repeated four times. The first column is the value of the MaxRetrans parameter. The second column gives the number of nodes in $RG_{(1,MR)}$. The third column gives the difference between the number of nodes for MaxRetrans in column 1 of the current row and the number of nodes in the previous row, i.e. for row 2 we have 92 nodes in column 2 and for row 1 we have 12 nodes in column 2. Hence the first entry in column 3 is $92 - 12 = 80$.

From Table C.2, after taking successive differences four times, we reach a point where the difference is a constant, 40, in column 6. Hence, we are able to find a quartic polynomial in MaxRetrans that fits this data. Let $|V_{(MS,MR)}| = aMR^4 + bMR^3 + cMR^2 + dMR + e$. Taking $MR = 0$ gives

\[
12 = a \cdot 0 + b \cdot 0 + c \cdot 0 + d \cdot 0 + e
\]

\[
12 = e \quad \text{(C.1)}
\]
Table C.2: Successive differences in the number of nodes when MaxSeqNo = 1.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Nodes</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>92</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>336</td>
<td>244</td>
<td>164</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>880</td>
<td>544</td>
<td>300</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1900</td>
<td>1020</td>
<td>476</td>
<td>176</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>3612</td>
<td>1712</td>
<td>692</td>
<td>216</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>6272</td>
<td>2660</td>
<td>948</td>
<td>256</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>10176</td>
<td>3904</td>
<td>1244</td>
<td>296</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>15660</td>
<td>5484</td>
<td>1580</td>
<td>336</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>23100</td>
<td>7440</td>
<td>1956</td>
<td>376</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>32912</td>
<td>9812</td>
<td>2372</td>
<td>416</td>
<td>40</td>
</tr>
</tbody>
</table>

Taking $MR = 1$ gives

\[92 = a + b + c + d + e\]
\[80 = a + b + c + d \text{ (subtracting (C.1) to eliminate } e)\] (C.2)

Taking $MR = 2$ gives

\[336 = a.2^4 + b.2^3 + c.2^2 + d.2 + e\]
\[324 = 16a + 8b + 4c + 2d \text{ (subtracting (C.1) to eliminate } e)\]
\[162 = 8a + 4b + 2c + d\]
\[82 = 7a + 3b + c \text{ (subtracting (C.2) to eliminate } d)\] (C.3)

Taking $MR = 3$ gives

\[880 = a.3^4 + b.3^3 + c.3^2 + d.3 + e\]
\[868 = 81a + 27b + 9c + 3d \text{ (subtracting (C.1) to eliminate } e)\]
\[628 = 78a + 24b + 6c \text{ (subtracting 3 times (C.2) to eliminate } d)\]
\[314 = 39a + 12b + 3c\]
\[68 = 18a + 3b \text{ (subtracting 3 times (C.3) to eliminate } c)\] (C.4)
C.2. The Size of the Reachability Graph

Table C.3: Successive differences in the number of nodes when MaxSeqNo = 2.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Nodes</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>138</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>504</td>
<td>366</td>
<td>246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1320</td>
<td>816</td>
<td>450</td>
<td>204</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2850</td>
<td>1530</td>
<td>714</td>
<td>264</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>5418</td>
<td>2568</td>
<td>1038</td>
<td>324</td>
<td>60</td>
</tr>
<tr>
<td>6</td>
<td>9408</td>
<td>3990</td>
<td>1422</td>
<td>384</td>
<td>60</td>
</tr>
<tr>
<td>7</td>
<td>15264</td>
<td>5856</td>
<td>1866</td>
<td>444</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>23490</td>
<td>8226</td>
<td>2370</td>
<td>504</td>
<td>60</td>
</tr>
<tr>
<td>9</td>
<td>34650</td>
<td>11160</td>
<td>2934</td>
<td>564</td>
<td>60</td>
</tr>
<tr>
<td>10</td>
<td>49368</td>
<td>14718</td>
<td>3558</td>
<td>624</td>
<td>60</td>
</tr>
</tbody>
</table>

Taking \( MR = 4 \) gives

\[
1900 = a4^4 + b4^3 + c4^2 + d.4 + e
\]

\[
1888 = 256a + 64b + 16c + 4d \text{ (subtracting (C.1) to eliminate } e)\]

\[
472 = 64a + 16b + 4c + d
\]

\[
392 = 63a + 15b + 3c \text{ (subtracting (C.2) to eliminate } d)\]

\[
146 = 42a + 6b \text{ (subtracting 3 times (C.3) to eliminate } c)\]

\[
73 = 21a + 3b
\]

\[
5 = 3a \text{ (subtracting (C.4) to eliminate } b) \quad \text{(C.5)}
\]

Hence \( a = 5/3 \) from (C.5). Substituting this into (C.4) gives \( b = (68 - 18 \times (5/3)) = 38/3 \). Substituting \( a \) and \( b \) into (C.3) gives \( c = (82 - 7 \times (5/3) - 3 \times (38/3)) = 97/3 \). Substituting \( a, b \) and \( c \) into (C.2) gives \( d = (80 - 5/3 - 38/3 - 97/3) = 100/3 \). Hence,

\[
|V(1,MR)| = (1/3) \times (5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36)
\]

\[
|V(1,MR)| = (2/6) \times (5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36) \quad \text{(C.6)}
\]

The case where MaxSeqNo = 2

Following the same procedure for MaxSeqNo = 2 from the data in Table C.3= gives:
C.2. The Size of the Reachability Graph

\[ 18 = a.0 + b.0 + c.0 + d.0 + e \]
\[ 18 = e \] \hspace{1cm} (C.7)

Taking \( MR = 1 \) gives

\[ 138 = a + b + c + d + e \]
\[ 120 = a + b + c + d \] \hspace{1cm} (subtracting (C.7) to eliminate \( e \)) \hspace{1cm} (C.8)

Taking \( MR = 2 \) gives

\[ 504 = a.2^4 + b.2^3 + c.2^2 + d.2 + e \]
\[ 486 = 16a + 8b + 4c + 2d \] \hspace{1cm} (subtracting (C.7) to eliminate \( e \)) \hspace{1cm} (C.8)
\[ 243 = 8a + 4b + 2c + d \]
\[ 123 = 7a + 3b + c \] \hspace{1cm} (subtracting (C.8) to eliminate \( d \)) \hspace{1cm} (C.9)

Taking \( MR = 3 \) gives

\[ 1320 = a.3^4 + b.3^3 + c.3^2 + d.3 + e \]
\[ 1302 = 81a + 27b + 9c + 3d \] \hspace{1cm} (subtracting (C.7) to eliminate \( e \)) \hspace{1cm} (C.8)
\[ 942 = 78a + 24b + 6c \] \hspace{1cm} (subtracting 3 times (C.8) to eliminate \( d \)) \hspace{1cm} (C.8)
\[ 471 = 39a + 12b + 3c \] \hspace{1cm} (subtracting (C.8) to eliminate \( d \)) \hspace{1cm} (C.9)
\[ 102 = 18a + 3b \] \hspace{1cm} (subtracting 3 times (C.9) to eliminate \( c \)) \hspace{1cm} (C.10)

Taking \( MR = 4 \) gives

\[ 2850 = a.4^4 + b.4^3 + c.4^2 + d.4 + e \]
\[ 2832 = 256a + 64b + 16c + 4d \] \hspace{1cm} (subtracting (C.7) to eliminate \( e \)) \hspace{1cm} (C.8)
\[ 708 = 64a + 16b + 4c + d \]
\[ 588 = 63a + 15b + 3c \] \hspace{1cm} (subtracting (C.8) to eliminate \( d \)) \hspace{1cm} (C.9)
\[ 219 = 42a + 6b \] \hspace{1cm} (subtracting 3 times (C.9) to eliminate \( c \)) \hspace{1cm} (C.10)
\[ 15 = 6a \] \hspace{1cm} (subtracting 2 times (C.10) to eliminate \( b \)) \hspace{1cm} (C.11)

Hence \( a = 15/6 = 5/2 \) from (C.11). Substituting this into (C.10) gives \( b = (102 - 18 * (15/6))/3 = 57/3 = 38/2 \). Substituting \( a \) and \( b \) into (C.9) gives \( c = (123 - 7 * (5/2) - 3 * (38/2)) = 97/2 \). Substituting \( a \), \( b \) and \( c \) into (C.8) gives \( d = (120 - 5/2 - 38/2 - 97/2) = 100/2 \). Hence,

\[ |V_{(2, MR)}| = (1/2) * (5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36) \]
\[ |V_{(2, MR)}| = (3/6) * (5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36) \] \hspace{1cm} (C.12)
C.2. The Size of the Reachability Graph

Table C.4: Successive differences in the number of nodes when MaxSeqNo = 3.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Nodes</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>184</td>
<td>160</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>672</td>
<td>488</td>
<td>328</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1760</td>
<td>1088</td>
<td>600</td>
<td>272</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3800</td>
<td>2040</td>
<td>952</td>
<td>352</td>
<td>80</td>
</tr>
<tr>
<td>5</td>
<td>7224</td>
<td>3424</td>
<td>1384</td>
<td>432</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>12544</td>
<td>5320</td>
<td>1896</td>
<td>512</td>
<td>80</td>
</tr>
<tr>
<td>7</td>
<td>20352</td>
<td>7808</td>
<td>2488</td>
<td>592</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>31320</td>
<td>10968</td>
<td>3160</td>
<td>672</td>
<td>80</td>
</tr>
<tr>
<td>9</td>
<td>46200</td>
<td>14880</td>
<td>3912</td>
<td>752</td>
<td>80</td>
</tr>
<tr>
<td>10</td>
<td>65824</td>
<td>19624</td>
<td>4744</td>
<td>832</td>
<td>80</td>
</tr>
</tbody>
</table>

The case where MaxSeqNo = 3

Following the same procedure for MaxSeqNo=3 from the data in Table C.4, (omitting the details) gives

\[ |V_{(3;MR)}| = (2/3) \times (5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36) \]

\[ |V_{(3;MR)}| = (4/6) \times (5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36) \quad (C.13) \]

The case where MaxSeqNo = 4

Following the same procedure for MaxSeqNo=4 from the data in Table C.5, (omitting the details) gives

\[ |V_{(4;MR)}| = (5/6) \times (5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36) \quad (C.14) \]

The case where MaxSeqNo = 5

Following the same procedure for MaxSeqNo=5 from the data in Table C.6, (omitting the details) gives

\[ |V_{(5;MR)}| = 5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36 \]

\[ |V_{(5;MR)}| = (6/6)(5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36) \quad (C.15) \]

Factoring out MaxRetrans

In all of Equations (C.6) (MS = 1), (C.12) (MS = 2), (C.13) (MS = 3), (C.14) (MS = 4) and (C.15) (MS = 5), the same expression in MR appears: \(5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36\). Hence,
### C.2. The Size of the Reachability Graph

Table C.5: Successive differences in the number of nodes when $\text{MaxSeqNo} = 4$.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Nodes</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>230</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>840</td>
<td>610</td>
<td>410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2200</td>
<td>1360</td>
<td>750</td>
<td>340</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4750</td>
<td>2550</td>
<td>1190</td>
<td>440</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>9030</td>
<td>4280</td>
<td>1730</td>
<td>540</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>15680</td>
<td>6650</td>
<td>2370</td>
<td>640</td>
<td>100</td>
</tr>
<tr>
<td>7</td>
<td>25440</td>
<td>9760</td>
<td>3110</td>
<td>740</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>39150</td>
<td>13710</td>
<td>3950</td>
<td>840</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>57750</td>
<td>18600</td>
<td>4890</td>
<td>940</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>82280</td>
<td>24530</td>
<td>5930</td>
<td>1040</td>
<td>100</td>
</tr>
</tbody>
</table>

Table C.6: Successive differences in the number of nodes when $\text{MaxSeqNo} = 5$.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Nodes</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>276</td>
<td>240</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1008</td>
<td>732</td>
<td>492</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2640</td>
<td>1632</td>
<td>900</td>
<td>408</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5700</td>
<td>3060</td>
<td>1428</td>
<td>528</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>10836</td>
<td>5136</td>
<td>2076</td>
<td>648</td>
<td>120</td>
</tr>
<tr>
<td>6</td>
<td>18816</td>
<td>7980</td>
<td>2844</td>
<td>768</td>
<td>120</td>
</tr>
<tr>
<td>7</td>
<td>30528</td>
<td>11712</td>
<td>3732</td>
<td>888</td>
<td>120</td>
</tr>
<tr>
<td>8</td>
<td>46980</td>
<td>16452</td>
<td>4740</td>
<td>1008</td>
<td>120</td>
</tr>
<tr>
<td>9</td>
<td>69300</td>
<td>22320</td>
<td>5868</td>
<td>1128</td>
<td>120</td>
</tr>
<tr>
<td>10</td>
<td>98736</td>
<td>29436</td>
<td>7116</td>
<td>1248</td>
<td>120</td>
</tr>
</tbody>
</table>
C.2. The Size of the Reachability Graph

Table C.7: Successive differences in the number of arcs when \( \text{MaxSeqNo} = 1 \).

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Arcs</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>242</td>
<td>230</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1166</td>
<td>924</td>
<td>694</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3494</td>
<td>2328</td>
<td>1404</td>
<td>710</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8176</td>
<td>4682</td>
<td>2354</td>
<td>950</td>
<td>240</td>
</tr>
<tr>
<td>5</td>
<td>16402</td>
<td>8226</td>
<td>3544</td>
<td>1190</td>
<td>240</td>
</tr>
<tr>
<td>6</td>
<td>29602</td>
<td>13200</td>
<td>4974</td>
<td>1430</td>
<td>240</td>
</tr>
<tr>
<td>7</td>
<td>49446</td>
<td>19844</td>
<td>6644</td>
<td>1670</td>
<td>240</td>
</tr>
<tr>
<td>8</td>
<td>77844</td>
<td>28398</td>
<td>8554</td>
<td>1910</td>
<td>240</td>
</tr>
<tr>
<td>9</td>
<td>116946</td>
<td>39102</td>
<td>10704</td>
<td>2150</td>
<td>240</td>
</tr>
<tr>
<td>10</td>
<td>169142</td>
<td>52196</td>
<td>13094</td>
<td>2390</td>
<td>240</td>
</tr>
</tbody>
</table>

Evidence suggests that the number of nodes in the RG is quartic in \( \text{MaxRetrans} \), given by the formula 
\[
|V_{(MS,MR)}| = (\text{multiplier}) \times (5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36).
\]
For \( MS = 1 \), multiplier = 2/6. For \( MS = 2 \), multiplier = 3/6. For \( MS = 3 \), multiplier = 4/6. For \( MS = 4 \), multiplier = 5/6.
For \( MS = 5 \), multiplier = 6/6. Thus, evidence suggests that the multiplier is a function of \( MS \), namely \((MS + 1)/6\).

Thus we conjecture that 
\[
|V_{(MS,MR)}| = ((MS+1)/6) \times (5MR^4 + 38MR^3 + 97MR^2 + 100MR + 36).
\]

C.2.2 The Number of Arcs

To determine a polynomial for the number of arcs, we follow the same procedure that was followed to determine a formula for the number of nodes.

The case where \( \text{MaxSeqNo} = 1 \)

Table C.7 contains the successive difference in the number of arcs repeated four times. The columns of this table give the same information as for the nodes, with the only difference being that the second column gives the number of arcs in \( RG_{(1,MR)} \) rather than the number of nodes.

From Table C.7, after taking successive differences four times, we reach a point where the difference is a constant, 240, in column 6. Hence, we are able to find a quartic polynomial in \( \text{MaxRetrans} \) that fits
C.2. The Size of the Reachability Graph

this data. Let $|A_{(MS,MR)}| = aMR^4 + bMR^3 + cMR^2 + dMR + e$. Taking $MR = 0$ gives

$$12 = a.0 + b.0 + c.0 + d.0 + e$$

$$12 = e \quad \text{(C.16)}$$

Taking $MR = 1$ gives

$$242 = a + b + c + d + e$$
$$230 = a + b + c + d \quad \text{(subtracting (C.16) to eliminate $e$)} \quad \text{(C.17)}$$

Taking $MR = 2$ gives

$$1166 = a.2^4 + b.2^3 + c.2^2 + d.2 + e$$
$$1154 = 16a + 8b + 4c + 2d \quad \text{(subtracting (C.16) to eliminate $e$)}$$
$$577 = 8a + 4b + 2c + d$$
$$347 = 7a + 3b + c \quad \text{(subtracting (C.17) to eliminate $d$)} \quad \text{(C.18)}$$

Taking $MR = 3$ gives

$$3494 = a.3^4 + b.3^3 + c.3^2 + d.3 + e$$
$$3482 = 81a + 27b + 9c + 3d \quad \text{(subtracting (C.16) to eliminate $e$)}$$
$$2792 = 78a + 24b + 6c \quad \text{(subtracting 3 times (C.17) to eliminate $d$)}$$
$$1396 = 39a + 12b + 3c$$
$$355 = 18a + 3b + c \quad \text{(subtracting 3 times (C.18) to eliminate $c$)} \quad \text{(C.19)}$$

Taking $MR = 4$ gives

$$8176 = a.4^4 + b.4^3 + c.4^2 + d.4 + e$$
$$8164 = 256a + 64b + 16c + 4d \quad \text{(subtracting (C.16) to eliminate $e$)}$$
$$2041 = 64a + 16b + 4c + d$$
$$1811 = 63a + 15b + 3c \quad \text{(subtracting (C.17) to eliminate $d$)}$$
$$770 = 42a + 6b \quad \text{(subtracting 3 times (C.18) to eliminate $c$)}$$
$$385 = 21a + 3b$$
$$30 = 3a \quad \text{(subtracting (C.19) to eliminate $b$)} \quad \text{(C.20)}$$

Hence $a = 10$ from (C.20). Substituting this into (C.19) gives $b = (355 - 18 * 10)/3 = 175/3$. Substituting $a$ and $b$ into (C.18) gives $c = (347 - 7 * 10 - 3 * (175/3)) = 102$. Substituting $a$, $b$ and $c$
C.2. The Size of the Reachability Graph

Table C.8: Successive differences in the number of arcs when $\text{MaxSeqNo} = 2$.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Arcs</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>363</td>
<td>345</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1749</td>
<td>1386</td>
<td>1041</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5241</td>
<td>3492</td>
<td>2106</td>
<td>1065</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12264</td>
<td>7023</td>
<td>3531</td>
<td>1425</td>
<td>360</td>
</tr>
<tr>
<td>5</td>
<td>24603</td>
<td>12339</td>
<td>5316</td>
<td>1785</td>
<td>360</td>
</tr>
<tr>
<td>6</td>
<td>44403</td>
<td>19800</td>
<td>7461</td>
<td>2145</td>
<td>360</td>
</tr>
<tr>
<td>7</td>
<td>74169</td>
<td>29766</td>
<td>9966</td>
<td>2505</td>
<td>360</td>
</tr>
<tr>
<td>8</td>
<td>116766</td>
<td>42597</td>
<td>12831</td>
<td>2865</td>
<td>360</td>
</tr>
<tr>
<td>9</td>
<td>175419</td>
<td>58653</td>
<td>16056</td>
<td>3225</td>
<td>360</td>
</tr>
<tr>
<td>10</td>
<td>253713</td>
<td>78294</td>
<td>19641</td>
<td>3585</td>
<td>360</td>
</tr>
</tbody>
</table>

Into (C.17) gives $d = (230 - 10 - 175/3 - 102) = 179/3$. Hence,

$$|A_{(1, MR)}| = (1/3) * (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)$$

$$|A_{(1, MR)}| = (2/6) * (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)$$

(C.21)

The case where $\text{MaxSeqNo} = 2$

Following the same procedure for $\text{MaxSeqNo} = 2$ from the data in Table C.8 gives:

$$18 = a.0 + b.0 + c.0 + d.0 + e$$

$$18 = e$$

(C.22)

Taking $MR = 1$ gives

$$363 = a + b + c + d + e$$

$$345 = a + b + c + d \ (\text{subtracting (C.22) to eliminate } e)$$

(C.23)

Taking $MR = 2$ gives

$$1749 = a.2^4 + b.2^3 + c.2^2 + d.2 + e$$

$$1731 = 16a + 8b + 4c + 2d \ (\text{subtracting (C.22) to eliminate } e)$$

$$1041 = 14a + 6b + 2c \ (\text{subtracting 2 times (C.23) to eliminate } d)$$

(C.24)
C.2. The Size of the Reachability Graph

Taking $MR = 3$ gives

$$5241 = a \cdot 3^4 + b \cdot 3^3 + c \cdot 3^2 + d \cdot 3 + e$$

$$5223 = 81a + 27b + 9c + 3d \text{ (subtracting (C.22) to eliminate } e \text{)}$$

$$4188 = 78a + 24b + 6c \text{ (subtracting 3 times (C.23) to eliminate } d \text{)}$$

$$1065 = 36a + 6b \text{ (subtracting 3 times (C.24) to eliminate } c \text{)}$$

(C.25)

Taking $MR = 4$ gives

$$12264 = a \cdot 4^4 + b \cdot 4^3 + c \cdot 4^2 + d \cdot 4 + e$$

$$12246 = 256a + 64b + 16c + 4d \text{ (subtracting (C.22) to eliminate } e \text{)}$$

$$6123 = 128a + 32b + 8c + 2d$$

$$5433 = 126a + 30b + 6c \text{ (subtracting 2 times (C.23) to eliminate } d \text{)}$$

$$2310 = 84a + 12b \text{ (subtracting 3 times (C.24) to eliminate } c \text{)}$$

$$180 = 12a \text{ (subtracting 2 times (C.25) to eliminate } b \text{)}$$

(C.26)

Hence $a = 15$ from (C.26). Substituting this into (C.25) gives $b = (1065 - 36 \cdot 15)/6 = 175/2$.

Substituting $a$ and $b$ into (C.24) gives $c = (1041 - 14 \cdot 15 - 6 \cdot (175/2))/2 = 306/2$. Substituting $a$, $b$, and $c$ into (C.23) gives $d = (345 - 15 - 175/2 - 306/2) = 179/2$.

Hence,

$$|A_{(2,MR)}| = (1/2) \cdot (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)$$

$$|A_{(2,MR)}| = (3/6) \cdot (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)$$

(C.27)

The case where $\text{MaxSeqNo} = 3$

Following the same procedure for $\text{MaxSeqNo}=3$ from the data in Table C.9, (omitting the details) gives

$$|A_{(3,MR)}| = (2/3) \cdot (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)$$

$$|A_{(3,MR)}| = (4/6) \cdot (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)$$

(C.28)

The case where $\text{MaxSeqNo} = 4$

Following the same procedure for $\text{MaxSeqNo}=4$ from the data in Table C.10, (omitting the details) gives

$$|A_{(4,MR)}| = (5/6) \cdot (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36)$$

(C.29)
### C.2. The Size of the Reachability Graph

Table C.9: Successive differences in the number of arcs when MaxSeqNo = 3.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Arcs</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>484</td>
<td>460</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2332</td>
<td>1848</td>
<td>1388</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6988</td>
<td>4656</td>
<td>2808</td>
<td>1420</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16352</td>
<td>9364</td>
<td>4708</td>
<td>1900</td>
<td>480</td>
</tr>
<tr>
<td>5</td>
<td>32804</td>
<td>16452</td>
<td>7088</td>
<td>2380</td>
<td>480</td>
</tr>
<tr>
<td>6</td>
<td>59204</td>
<td>26400</td>
<td>9948</td>
<td>2860</td>
<td>480</td>
</tr>
<tr>
<td>7</td>
<td>98892</td>
<td>39688</td>
<td>13288</td>
<td>3340</td>
<td>480</td>
</tr>
<tr>
<td>8</td>
<td>155688</td>
<td>56796</td>
<td>17108</td>
<td>3820</td>
<td>480</td>
</tr>
<tr>
<td>9</td>
<td>233892</td>
<td>78204</td>
<td>21408</td>
<td>4300</td>
<td>480</td>
</tr>
<tr>
<td>10</td>
<td>338284</td>
<td>104392</td>
<td>26188</td>
<td>4780</td>
<td>480</td>
</tr>
</tbody>
</table>

Table C.10: Successive differences in the number of arcs when MaxSeqNo = 4.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Arcs</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>605</td>
<td>575</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2915</td>
<td>2310</td>
<td>1735</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8735</td>
<td>5820</td>
<td>3510</td>
<td>1775</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20440</td>
<td>11705</td>
<td>5885</td>
<td>2375</td>
<td>600</td>
</tr>
<tr>
<td>5</td>
<td>41005</td>
<td>20565</td>
<td>8860</td>
<td>2975</td>
<td>600</td>
</tr>
<tr>
<td>6</td>
<td>74005</td>
<td>33000</td>
<td>12435</td>
<td>3575</td>
<td>600</td>
</tr>
<tr>
<td>7</td>
<td>123615</td>
<td>49610</td>
<td>16610</td>
<td>4175</td>
<td>600</td>
</tr>
<tr>
<td>8</td>
<td>194610</td>
<td>70995</td>
<td>21385</td>
<td>4775</td>
<td>600</td>
</tr>
<tr>
<td>9</td>
<td>292365</td>
<td>97755</td>
<td>26760</td>
<td>5375</td>
<td>600</td>
</tr>
<tr>
<td>10</td>
<td>422855</td>
<td>130490</td>
<td>32735</td>
<td>5975</td>
<td>600</td>
</tr>
</tbody>
</table>
C.2. The Size of the Reachability Graph

Table C.11: Successive differences in the number of arcs when MaxSeqNo = 5.

<table>
<thead>
<tr>
<th>MaxRetrans</th>
<th>No. of Arcs</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Cubic</th>
<th>Quartic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>726</td>
<td>690</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3498</td>
<td>2772</td>
<td>2082</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10482</td>
<td>6984</td>
<td>4212</td>
<td>2130</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>24528</td>
<td>14046</td>
<td>7062</td>
<td>2850</td>
<td>720</td>
</tr>
<tr>
<td>5</td>
<td>49206</td>
<td>24678</td>
<td>10632</td>
<td>3570</td>
<td>720</td>
</tr>
<tr>
<td>6</td>
<td>88806</td>
<td>39600</td>
<td>14922</td>
<td>4290</td>
<td>720</td>
</tr>
<tr>
<td>7</td>
<td>148338</td>
<td>59532</td>
<td>19932</td>
<td>5010</td>
<td>720</td>
</tr>
<tr>
<td>8</td>
<td>233532</td>
<td>85194</td>
<td>25662</td>
<td>5730</td>
<td>720</td>
</tr>
<tr>
<td>9</td>
<td>350838</td>
<td>117306</td>
<td>32112</td>
<td>6450</td>
<td>720</td>
</tr>
<tr>
<td>10</td>
<td>507426</td>
<td>156588</td>
<td>39282</td>
<td>7170</td>
<td>720</td>
</tr>
</tbody>
</table>

The case where MaxSeqNo = 5

Following the same procedure for MaxSeqNo=5 from the data in Table C.11, (omitting the details) gives

\[ |A_{(5,MR)}| = 30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36 \]
\[ |A_{(5,MR)}| = (6/6)(30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36) \]  

(C.30)

Factoring out MaxRetrans

In all of Equations (C.21) (MS = 1), (C.27) (MS = 2), (C.28) (MS = 3), (C.29) (MS = 4) and (C.30) (MS = 5), the same expression in MR appears: 30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36. Hence, evidence suggests that the number of nodes in the RG is quartic in MaxRetrans, given by the formula

\[ |A_{(MS,MR)}| = \text{(multiplier)} \times (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36). \]

For MS = 1, multiplier = 2/6. For MS = 2, multiplier = 3/6. For MS = 3, multiplier = 4/6. For MS = 4, multiplier = 5/6. For MS = 5, multiplier = 6/6. Thus, evidence suggests that the multiplier is a function of MS, namely (MS + 1)/6.

Thus we conjecture that

\[ |A_{(MS,MR)}| = ((MS+1)/6) \times (30MR^4 + 175MR^3 + 306MR^2 + 179MR + 36). \]
Appendix D

Publications

This appendix gives the citation and abstract of all journal papers, and refereed conference and workshop papers completed during my candidacy, not all of which are directly related to my PhD.

D.1 Journal Publications


   **Abstract.** The sweep-line state space method allows states to be deleted from memory during state exploration, thus alleviating the state explosion problem. Properties of the system (such as the absence of deadlocks) can then be verified on-the-fly. This paper presents an extension to the sweep-line method that allows on-the-fly checking of safety properties expressed as sequences of actions of the modelled system. This has been implemented in a prototype sweep-line library for Coloured Petri nets. We evaluate the prototype by applying it to the connection management procedures of the Datagram Congestion Control Protocol, a new Internet transport protocol.


   **Abstract.** The Stop-and-Wait protocol (SWP) has two (unbounded) parameters: the maximum sequence number (MaxSeqNo) and the maximum number of retransmissions (MaxRetrans). This paper presents an algebraic method for analysis of the SWP for all possible values of these parameters. Model checking
such a system requires considering an infinite family of models, one for each combination of parameter values, and thus an infinite family of state spaces (reachability graphs). These reachability graphs are represented symbolically by a set of algebraic formulas in MaxSeqNo and MaxRetrans. This result is significant as it provides a complete characterisation of the infinite set of reachability graphs of our SWP model in both parameters, allowing properties to be verified for the infinite class. Verification of a number of properties is described.


Abstract. Most protocols contain parameters, such as the maximum number of retransmissions in an error recovery protocol. These parameters are instantiated with values that depend on the operating environment of the protocol. We would therefore like our formal specification or model of the system to include these parameters symbolically, where in general each parameter will have an arbitrary upper limit. The inclusion of parameters produces infinite family of finite state systems, which makes verification difficult. However, techniques and tools are being developed for the verification of parametric and infinite state systems. We explore the use of one such tool, FAST, for automatically verifying several properties (such as channel bounds and the stop-and-wait property of alternating sends and receives) of the stop-and-wait class of protocols, where the maximum number of retransmissions and the maximum sequence number are considered as unbounded parameters. Coloured Petri nets (CPNs), an expressive language for representing protocols, are used to model this stop-and-wait class. However, FAST’s foundation is counter systems, automata where states are a vector of non-negative integers and with operations limited to Presburger arithmetic. We therefore also present some first steps in transforming CPNs to counter systems in the context of stop-and-wait protocols operating over unbounded FIFO channels.


Abstract. Despite the widespread use of the Transmission Control Protocol (TCP) as the main transport protocol in the Internet, the procedures for connection establishment and release are still not fully understood. This paper extends the analysis of a Coloured Petri net model of TCP’s Connection Management procedures by applying the state explosion alleviation technique known as the sweep-line method. The protocol is assumed to be operating over a reordering lossless channel. Termination and absence of dead-
lock properties are investigated for many scenarios, including client-server and simultaneous connection establishment, orderly release and abortion. The sweep-line method provides a reduction in memory usage of around a factor of 10 and allows investigation of many scenarios that were previously out of the reach of conventional methods.


**Abstract.** The Stop-and-Wait protocol (SWP) has two (unbounded) parameters: the maximum sequence number (MaxSeqNo) and the maximum number of retransmissions (MaxRetrans). Our aim is to verify this protocol for all possible values of these parameters. Model checking such a system requires considering an infinite family of state spaces (reachability graphs). We firstly show that the size of these state spaces is linear in MaxSeqNo and quartic in MaxRetrans. This leads us to develop a symbolic representation for the reachability graphs which can be viewed as a symbolic Finite State Automaton (FSA). We apply automata reduction techniques directly to the symbolic FSA to obtain a language equivalent FSA representing the sequences of externally visible events. This FSA is independent of the parameters. We confirm that this is language equivalent to the Stop-and-Wait service of alternating send and receive events. The results are significant as we have: 1. a novel algebraic representation of the infinite set of reachability graphs and their related FSAs of our SWP model; and 2. verified conformance of the SWP to its service, for all values of the unbounded MaxSeqNo parameter.


**Abstract.** The correct operation of communication and co-operation protocols, including signalling systems in various networks, is essential for the reliability of the many distributed systems that facilitate our global economy. This paper presents a methodology for the formal specification, analysis and verification of protocols based on the use of Coloured Petri Nets (CPNs) and automata theory. The methodology is illustrated using two case studies. The first belongs to the category of data transfer protocols, called Stop-and-Wait Protocols, while the second investigates the connection management part of the Internet’s Transmission Control Protocol (TCP). Stop-and-Wait Protocols (SWP) incorporate retransmission strategies to recover from data transmission errors that occur on noisy transmission media. Although relatively simple, their basic mechanisms are important for practical protocols such as the data transfer procedures.
of TCP. The SWP case study is quite detailed. It considers a class of protocols characterized by two parameters: the maximum sequence number ($\text{MaxSeqNo}$) and the maximum number of retransmissions ($\text{MaxRetrans}$). We investigate the operation of the protocol over (lossy) in-sequence (FIFO) channels, and then over (lossy) re-ordering media, such as that provided by the Internet Protocol. Four properties are considered: the bound on the number of messages that can be in the communication channels; whether or not the protocol provides the expected service of alternating sends and receives; (unknown) loss of messages (i.e. data sent but not received, and not detected as lost by the protocol); and the acceptance of duplicates as new messages. The model is analysed using a combination of hand proofs and automatic techniques. A new result for the bound of the channels ($2\text{MaxRetrans}+1$) is proved for FIFO channels. It is further shown that for re-ordering channels, the channels are unbounded, loss and duplication can occur, and that the SWP does not provide the expected service. We discuss the relevance of these results to the Transmission Control Protocol and indicate the limitations of our approach and the need for further work. The second case study (TCP) illustrates the use of hierarchies to provide a compact and readable CPN model for a complex protocol. We advocate an incremental approach to both modelling and analysis. The importance of stating the assumptions involved is emphasised and we illustrate how they affect the abstractions that can be made to simplify the model. The incremental approach to analysis allows us to validate the model against the TCP definition and to show how errors in the connection establishment procedures can be found. Finally we provide some observations and tips on the how the methodology can be used based on many years of experience. The emphasis of the paper is on providing a tutorial style introduction to the methodology, examining case studies in depth, rather than breadth, and giving some insight into the process while noting its limitations.


Abstract. State space exploration is one of the main approaches to computer-aided verification and analysis of finite-state systems. It is used to reason about a wide range of properties during the design phase of a system, including system deadlocks. Unfortunately, state space exploration needs to handle huge state spaces for most practical systems. Several state space reduction methods have been developed to tackle this problem. In this paper we develop algorithms for combining two of these methods: state equivalence class reduction and the sweep-line. The algorithms allow deadlocks to be detected by recording terminal states of the system on-the-fly during state space exploration. We derive expressions for the complexity of the algorithms and demonstrate their usefulness with an industrial case study. Our results
show that the combined method achieves at least a 6 fold reduction of the state space for interesting parameter values compared with either method used in isolation, while still proving the desired system property of the terminal states. The runtime performance of the combined method is almost the same as that of the equivalence class method over the chosen parameter range. Moreover, the improvement in space reduction increases with increased parameter values.


Abstract. 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 invited paper revisits the class of Stop-and-Wait protocols that incorporate retransmission strategies to recover from transmission errors. This is motivated by the fact that their basic mechanisms are important for practical protocols such as the Internet’s Transmission Control Protocol (TCP). Stop-and-Wait protocols have been shown to operate correctly over media that may lose packets, however, there has been little discussion regarding the operation of these protocols over media that can re-order packets. The paper presents an investigation of these protocols operating over a medium, such as that provided by the Internet Protocol, that does allow reordering of data. Coloured Petri Nets are used to build a model of a Stop-and-Wait Protocol parameterized by its maximum sequence number and the maximum value of the retransmission counter. The model is analysed using a combination of hand proofs and automatic techniques. We identify four problems. We firstly prove the counter intuitive property for a Stop-and-Wait protocol that the number of packets that are stored in the network can grow without bound. This is true for any positive values of the maximum sequence number and the maximum number of retransmissions. We further show that loss of packets is possible and that duplicates can be accepted as new packets by the receiver. These first three properties hold even though the sender and receiver perceive that the protocol is operating correctly. The final problem is that the protocol does not satisfy the Stop-and-Wait service where sends and receives alternate. Finally, we provide a discussion of the relevance of these results to the Transmission Control Protocol.

D.2 Refereed Conference and Workshop Papers

Abstract. Military logistics concerns the activities required to support operational forces. It encompasses the storage and distribution of materiel, management of personnel and the provision of facilities and services. A desire to improve the efficiency and effectiveness of the Australian Defence Force logistics process has led to the investigation of rigorous military logistics models suitable for analysis and experimentation. Logistics networks can be viewed as distributed discrete event systems, and hence can be formalised with discrete event techniques which support concurrency. This paper presents a Coloured Petri Net (CPN) model of a military logistics system and discusses some of our experience in developing an initial model. Interesting modelling problems encountered, and their solutions and impact on CPN support tools, are discussed.


Abstract. State space explosion is a key problem in the analysis of finite state systems. The sweep-line method is a state exploration method which uses a notion of progress to allow states to be deleted from memory when they are no longer required. This reduces the peak number of states that need to be stored, while still exploring the full state space. The technique shows promise but has never achieved reductions greater than about a factor of 10 in the number of states stored in memory for industrially relevant examples. This paper discusses sweep-line analysis of the connection management procedures of a new Internet standard, the Datagram Congestion Control Protocol (DCCP). As the intuitive approaches to sweep-line analysis are not effective, we introduce new variables to track progress. This creates further state explosion. However, when used with the sweep-line, the peak number of states is reduced by over two orders of magnitude compared with the original. Importantly, this allows DCCP to be analysed for larger parameter values.


Abstract. This paper shows how a formal method in the form of Coloured Petri Nets (CPNs) and the
supporting CPN Tools have been used in the development of the Course of Action Scheduling Tool (COAST). The aim of COAST is to support human planners in the specification and scheduling of tasks in a Course of Action. CPNs have been used to develop a formal model of the task execution framework underlying COAST. The CPN model has been extracted in executable form from CPN Tools and embedded directly into COAST, thereby automatically bridging the gap between the formal specification and its implementation. The scheduling capabilities of COAST are based on state space exploration of the embedded CPN model. Planners interact with COAST using a domain-specific graphical user interface (GUI) that hides the embedded CPN model and analysis algorithms. This means that COAST is based on a rigorous semantical model, but the use of formal methods is transparent to the users. Trials of operational planning using COAST have been conducted within the Australian Defence Force.


Abstract. The sweep-line state space method allows states to be deleted from memory during state exploration, thus alleviating state explosion. Properties of the system (such as the absence of deadlocks) can then be verified on-the-fly. This paper presents an extension to the sweep-line method that allows on-the-fly checking of language inclusion, which is useful for protocol verification. This has been implemented in a prototype Sweep-line library for Design/CPN. We evaluate the prototype by applying it to the connection management procedures of the Datagram Congestion Control Protocol, a new Internet transport protocol.


Abstract. Protocols may contain parameters that are chosen from a wide range. In some cases we would like our analysis results to apply to an arbitrary upper limit on a parameter value, such as the maximum number of retransmissions. In this case we have an infinite family of finite state systems. This makes their verification difficult. However, techniques and tools are being developed for the verification of parametric and infinite state systems. We explore the use of one such tool, FAST, for verifying several properties of the stop-and-wait class of protocols, where the maximum number of retransmissions and
the maximum sequence number are considered parameters. We are also interested in using expressive languages for representing protocols such as Coloured Petri nets (CPNs). Fast’s foundation is counter systems, which are automata whose states are a vector of non-negative integers, with operations limited to Presburger arithmetic. We therefore also present some first steps in transforming CPNs to counter systems in the context of stop-and-wait protocols operating over unbounded FIFO channels.


Abstract. The sweep-line occurrence graph method exploits a behavioural notion of progress found in many systems. This allows states to be deleted that will not be revisited during occurrence graph generation, allowing fewer states to be stored in main memory for the necessary comparisons, thus providing savings in both memory and time. Properties of the system (such as deadlocks) can then be verified on-the-fly. This method is relatively new and needs to be evaluated on a range of examples. One class of protocols that seems to be suited to sweep-line analysis is transaction protocols. This is because transaction protocols often have an occurrence graph that starts with a request and finishes with the request being satisfied (or not). Thus there is a natural progression of states as the transaction proceeds. This paper provides insight into how to design a progress mapping, central to the use of the sweep-line method, for a transaction protocol known as the Internet Open Trading Protocol (IOTP). IOTP is quite complex and is modelled using hierarchical Coloured Petri Nets (CPNs). The sweep-line method is particularised for CPNs and three progress mappings are developed for IOTP. The results show that naive choices for the progress mapping leads to unnecessary regeneration of states, due to the mapping not being monotonic. Refinement of the mapping leads to a monotonic progress measure, which allows results to be obtained for IOTP that were not previously possible.


Abstract. The basic idea of the sweep-line state space method is to exploit a formal notion of progress found in many concurrent and distributed systems. Exploiting progress makes it possible to sweep through the state space of a CP-net while storing only a small fragment of the states in memory at
any time. Properties of the system can then be verified on-the-fly during the sweep of the state space. This can lead to significant savings in peak memory usage and computation time. Examples of systems possessing progress are transport protocols, transactions protocols, workflow models, and systems modelled with Timed CP-nets. We present SWEEP/CPN, a library extension to Design/CPN supporting the sweep-line method, and demonstrate its use.


Abstract. The basic idea of the sweep-line state space method is to exploit a formal notion of progress found in many concurrent and distributed systems. Exploiting progress makes it possible to sweep through the state space of a CP-net while storing only a small fragment of the states in memory at any time. Properties of the system can then be verified on-the-fly during the sweep of the state space. This can lead to significant savings in peak memory usage and computation time. Examples of systems possessing progress are transport protocols, transactions protocols, workflow models, and systems modelled with Timed CP-nets. We present SWEEP/CPN, a library extension to Design/CPN supporting the sweep-line method, and demonstrate its use.