AUGMENTING PETRI NETS TO MODEL HEALTH-CARE PROTOCOLS

by

SARAH-JANE WHITTAKER

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen’s University
Kingston, Ontario, Canada
September 2011

Copyright © Sarah-Jane Whittaker, 2011
For Ben, who saw me at my worst and never even blinked.
Abstract

An outbreak of an infectious illness can have a devastating impact on a population. Once confirmed, local health care organizations will attempt to reduce the spread of the disease by adopting a set of pre-defined guidelines. Modelling such a system presents a number of unique challenges: timing and probability constraints must be captured, scaling must be seamless and methods for analysis must be robust and efficient. To satisfy these requirements, an augmented form of Petri net known as a choice-point net is introduced in this thesis. In this data structure, timing is associated with event-based transitions that may fire multiple times to simulate the same event occurring several times in parallel. Events may result in several possible outcomes, or choices, each of which is given a probability of occurrence. A choice-point net may be scaled without requiring structural changes to the model and may be analyzed by unravelling it into a finite-state automaton representing (perhaps portions of) its behaviour. By translating questions about the protocol into the mathematical language of the net, recursive algorithms may then be employed to provide health-care professionals with answers to their questions. To demonstrate the expressiveness of choice-point nets, an actual, in-use protocol to control respiratory infection outbreaks in long-term care homes is modelled. Three similar abridged scenarios set in a small long-term care home are also modelled, analyzed and compared.
Acknowledgments

I would first like to thank my supervisors, Dr. Karen Rudie and Dr. James McLellan, for their patience and support during my PhD research. I would also like to recognize the contribution of Dr. Stefan Haar, whose perspective challenged me and improved choice-point nets as a result. Dr. Bob Tennent provided excellent advice during a crucial point in my research, which was very much appreciated. This work also benefited in some way from each of the members of the DES Research group: Anthony Auer, Behnam Behinaein, Lenko Grigorov, David Sears, Md Tawhid Bin Waez and Creag Winacott.

I have been fortunate to find many friends within the School of Computing: Dave Dove, Jiro Inoue, Irene Lafleche, Margaret Lamb, Richard Linley, Alan McLeod, Wendy Powley, Debby Robertson, Amber Simpson and Karolina Zurowska. Their insights, their humour and their help have proven invaluable. I am also grateful to my parents, Pauline Sicard and Dale Whittaker, for their love and encouragement throughout.

I would like to remember Clara Redmond and Mabel Walker, who inspired me and encouraged my academic pursuits from an early age.

Finally, I am infinitely grateful to my husband, Benjamin Hall, for his unwavering love and support. This thesis would not exist without him.
Contents

Abstract i

Acknowledgments ii

Contents iii

List of Tables vii

List of Figures viii

1 Introduction 1

2 Background 6

2.1 Petri Nets ........................................... 6
2.1.1 Marked Petri Nets ................................... 7
2.1.2 Labelled Petri Nets .................................. 10
2.1.3 Reachability Graphs .................................. 10
2.2 Discrete-Event Systems .......................... 11
2.2.1 Working With the Plant .......................... 12
2.2.2 Using Petri Nets .................................... 13
2.3 Mathematical Epidemiology ........................ 14
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>Disaster Response</td>
</tr>
<tr>
<td>3</td>
<td>Literature Review</td>
</tr>
<tr>
<td>3.1</td>
<td>Modelling Health-Care Response to a Disaster</td>
</tr>
<tr>
<td>3.2</td>
<td>Petri Nets and Health-Care Modelling</td>
</tr>
<tr>
<td>3.3</td>
<td>Further Related Health-Care Research</td>
</tr>
<tr>
<td>4</td>
<td>Choice-Point Nets and Augmented Reachability Graphs</td>
</tr>
<tr>
<td>4.1</td>
<td>A Small Example of a CN</td>
</tr>
<tr>
<td>4.2</td>
<td>Augmented Reachability Graphs</td>
</tr>
<tr>
<td>4.3</td>
<td>A Small Example of an ARG</td>
</tr>
<tr>
<td>4.4</td>
<td>ARG Construction</td>
</tr>
<tr>
<td>4.5</td>
<td>An Example of ARG Construction</td>
</tr>
<tr>
<td>5</td>
<td>Behavioural Equivalence</td>
</tr>
<tr>
<td>5.1</td>
<td>Algorithm 1</td>
</tr>
<tr>
<td>5.2</td>
<td>Algorithm 2</td>
</tr>
<tr>
<td>5.3</td>
<td>Algorithm 3</td>
</tr>
<tr>
<td>5.4</td>
<td>Algorithm 4</td>
</tr>
<tr>
<td>5.5</td>
<td>Algorithm 5</td>
</tr>
<tr>
<td>6</td>
<td>Question Restrictions</td>
</tr>
<tr>
<td>6.1</td>
<td>Applying Restrictions</td>
</tr>
<tr>
<td>6.2</td>
<td>Exploration Algorithms</td>
</tr>
<tr>
<td>6.3</td>
<td>Question Restrictions and Model Checking</td>
</tr>
<tr>
<td>7</td>
<td>Restricted Directed Acyclic Graphs</td>
</tr>
</tbody>
</table>
9.3 Scenarios and Questions .................................................. 159
9.4 Results ........................................................................... 162
  9.4.1 At Least Half of the Residents Become Infected .......... 162
  9.4.2 More Than Half of the Staff Become Infected ............. 163
  9.4.3 An Outbreak is Declared ............................................ 165

10 Conclusion ........................................................................ 167

Bibliography ......................................................................... 171
List of Tables

3.1 Summary of Discrete-Event Modelling Approaches (Part 1) . . . . . . 22
3.2 Summary of Discrete-Event Modelling Approaches (Part 2) . . . . . . 23

9.1 Analysis: At Least Half of the Residents Become Infected . . . . . . 162
9.2 Analysis: More Than Half of the Staff Become Infected . . . . . . . 164
9.3 Analysis: An Outbreak is Declared . . . . . . . . . . . . . . . . . . . 165
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>An Example of an Ordinary Petri Net</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>An Example of Sequential Firing of Transitions in an Ordinary Petri Net</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>An Example of a Petri Net and its Equivalent Finite-State Automaton</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>An Example of a Labelled Petri Net</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>An Example of a Reachability Graph</td>
<td>11</td>
</tr>
<tr>
<td>2.6</td>
<td>An Example of a Plant</td>
<td>12</td>
</tr>
<tr>
<td>2.7</td>
<td>The SIR Flow Chart</td>
<td>14</td>
</tr>
<tr>
<td>4.1</td>
<td>Choice-Point Net Representation of a Long-Term Care Home Protocol</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>Augmented Reachability Graph of the CN Model for the LTCH Protocol</td>
<td>39</td>
</tr>
<tr>
<td>7.1</td>
<td>Restricted Directed Acyclic Graph Generated From $A = {(\text{Isolate, \text{geq}, 2})}$ and $J = {(time, \text{gt}, 24)}$ for the LTCH Protocol</td>
<td>89</td>
</tr>
<tr>
<td>7.2</td>
<td>Restricted Directed Acyclic Graph Generated From $A = {(rI11, \text{eq}, 2), (time, \text{geq}, 48)}$ and $J = {(\text{Isolate, gt, 0})}$ for the LTCH Protocol</td>
<td>90</td>
</tr>
<tr>
<td>7.3</td>
<td>An Example of Conflict Graph Resolution (Part 1)</td>
<td>92</td>
</tr>
<tr>
<td>7.4</td>
<td>An Example of Conflict Graph Resolution (Part 2)</td>
<td>93</td>
</tr>
<tr>
<td>7.5</td>
<td>Walkthrough Conflict Graph After Construction</td>
<td>107</td>
</tr>
<tr>
<td>7.6</td>
<td>Walkthrough Conflict Graph After Resolution</td>
<td>109</td>
</tr>
<tr>
<td>Chapter</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>8.1</td>
<td>Susceptible Residents at the Common Meal</td>
<td>117</td>
</tr>
<tr>
<td>8.2</td>
<td>Susceptible and Ill Residents Interacting (No Potential or Declared Outbreak)</td>
<td>119</td>
</tr>
<tr>
<td>8.3</td>
<td>Susceptible and Ill Residents at Group Activities</td>
<td>120</td>
</tr>
<tr>
<td>8.4</td>
<td>Susceptible Staff Members Examining Unreported Ill Residents (No Potential or Declared Outbreak)</td>
<td>122</td>
</tr>
<tr>
<td>8.5</td>
<td>Ill Staff Members Examining Unreported Ill Residents</td>
<td>124</td>
</tr>
<tr>
<td>8.6</td>
<td>Ill Staff Members Examining Susceptible Residents (No Potential or Declared Outbreak)</td>
<td>125</td>
</tr>
<tr>
<td>8.7</td>
<td>Susceptible Staff Members Examining Reported Ill Residents (No Potential or Declared Outbreak)</td>
<td>126</td>
</tr>
<tr>
<td>8.8</td>
<td>Surveillance Data Being Submitted and Examined (No Potential or Declared Outbreak)</td>
<td>128</td>
</tr>
<tr>
<td>8.9</td>
<td>Susceptible Staff Members Going On And Off Duty During the Day</td>
<td>130</td>
</tr>
<tr>
<td>8.10</td>
<td>Unreported Ill Staff Members On And Off Duty</td>
<td>131</td>
</tr>
<tr>
<td>8.11</td>
<td>Reported Ill Staff Members Going On Duty</td>
<td>132</td>
</tr>
<tr>
<td>8.12</td>
<td>Reported Ill Staff Members Going Off Duty</td>
<td>133</td>
</tr>
<tr>
<td>8.13</td>
<td>Unreported Staff Members Monitoring Their Symptoms (No Potential or Declared Outbreak)</td>
<td>134</td>
</tr>
<tr>
<td>8.14</td>
<td>Ill Residents Recovering From the Cold (Part 1)</td>
<td>135</td>
</tr>
<tr>
<td>8.15</td>
<td>Ill Residents Recovering From the Cold (Part 2)</td>
<td>136</td>
</tr>
<tr>
<td>8.16</td>
<td>Staff Members Recovering From the Cold (Part 1)</td>
<td>137</td>
</tr>
<tr>
<td>8.17</td>
<td>Staff Members Recovering From the Cold (Part 2)</td>
<td>137</td>
</tr>
</tbody>
</table>
8.18 Supervisors Reporting Ill Staff Member Cases to the ICP (No Potential
or Declared Outbreak) ................................................. 140
8.19 ICP Reports Culminating in a Potential Outbreak ........................ 141
8.20 ICP Reports Culminating in a Declared Outbreak ...................... 145
8.21 ICP Determines Outbreak is not Possible ................................ 145
8.22 Declaring an Outbreak Over ........................................ 147

9.1 (Abridged) Susceptible Residents at the Common Meal .............. 151
9.2 (Abridged) Susceptible Staff Members Examining Unreported Ill Res-
idents ............................................................................. 153
9.3 (Abridged) Ill Staff Members Examining Unreported Ill Residents .. 154
9.4 (Abridged) Ill Staff Members Examining Susceptible Residents ... 155
9.5 (Abridged) Susceptible Staff Members Examining Reported Ill Residents155
9.6 (Abridged) Unreported Staff Members Monitoring Their Symptoms . 156
9.7 (Abridged) Ill Residents Recovering From the Cold .................... 156
9.8 (Abridged) Reports Culminating in a Declared Outbreak .............. 157
9.9 (Abridged) Declaring an Outbreak Over .................................... 158
Chapter 1

Introduction

The primary concern of any health-care system is to maintain the well-being of its population under both normal and adverse conditions. When emergencies arise, such as the outbreak of an infectious disease or contamination of the water supply, medical institutions will enact a set of guidelines or protocols to effectively contain the disease. There is no doubt that any procedure designed to manage an emergency is created with the best of intentions and meant to produce the best of outcomes. Ensuring that protocols actually yield these results has produced a wealth of research dedicated to modelling the impact of and response to disasters [12]. Indeed, when it comes to emergency planning, no one will argue the need for testing.

Existing modelling approaches commonly use simulation and systems of differential equations customized with a particular flavour (e.g., agent-based, stochastic). Although these methods provide predictions, they do not evaluate a variety of possible outcomes or their probability of occurrence. Informally put, it is typical to estimate what is likely to happen, not the likelihood of what could happen.

Specifically, the current body of research in this field suffers from three drawbacks.
The first is that existing modelling approaches typically provide point estimates for the measures of interest. Although it is possible to predict different variables within the model (e.g., the number of hospital beds occupied at time \( t \)), the results are single values (e.g., 36 hospital beds). The analysis rarely considers other possible outcomes or the probability of those outcomes (e.g., there is a 25% chance that fewer than 36 hospital beds are occupied at time \( t \)).

The second is that no approach which discussed the possibility of human failure could be located. These modelling techniques require the definition of baseline rates and/or assumptions for how people move through the system. Occasionally, distributions are employed to vary things within the model (e.g., time spent at a given workstation). None of the techniques take into account the possibility that people fail to do what is prescribed. In emergency situations, people will occasionally deviate from the plan due to any number of factors. It is vital to consider the impact of these actions on the system and the population under its care.

A third, and often overlooked, issue is the challenge of translating a protocol into a mathematical form. Procedures will be partly written in prose and will almost certainly contain graphical elements such as tables, figures and charts. The process could be documented in chronological order of events, or be divided according to individual responsibilities, or be broken up into “if-then” responses. Deriving a mathematical representation of such a document cannot be automated—it will be up to the designers to perform this task using their knowledge of the system. It is therefore vital that the modelling mechanism not be so complex that it makes capturing the protocol difficult. If the model is flawed, so will the analysis be flawed—the simplest modelling tool capable of capturing the system may prove a better choice than a complex
structure with more options that is difficult to use.

These concerns provide motivation for a new modelling approach which addresses these issues and satisfies five recommendations for disaster response models [12]:

\[
\ldots \text{health sector disaster response models should address real-world problems, be designed for maximum usability by response planners, strike the appropriate balance between simplicity and complexity, include appropriate outcomes that extend beyond those considered in traditional cost-effectiveness analyses and be designed to evaluate the many uncertainties inherent in disaster response.}
\]

The research contained in this thesis provides a mechanism for modelling and analysis that meets each of these recommendations, resulting in a new and (intended to be) intuitive approach that addresses existing concerns within the disaster planning community.

Emergency protocols are composed with respect to the actions and reactions of people within the system. For example, the appearance of influenza-like symptoms in at least two patients in a long-term care home requires testing to confirm the presence of such a strain. Discrete-event systems [16], which evolve as events occur, offer a means of representation and analysis. To ensure accuracy, the modelling structure must be able to represent timing, probability and a variety of resources. In addition, it should be at least somewhat intuitive for the sake of the modeler, who must translate a prose protocol into a mathematical model. Finally, the data structure must provide a means for designers of health-care protocols to ask a wide variety of questions and receive answers. For example, what is the likelihood that a certain percentage of the population will become infected? What is the probability that the same mistake will
be made repeatedly?

To satisfy the aforementioned modelling requirements and provide an approach to analysis that can answer health-care administrators’ questions, this thesis introduces choice-point nets (CNs), an augmented form of Petri nets, as the means to model a health-care system and protocol. Timing is attached to events via the net’s transitions by assigning a set of timers or a specified lapse. The firing of a transition is analogous to the occurrence of an event; firing may only occur if the transition is enabled and its timer matches the current time on the global clock or its lapse has expired. Events are allowed different outcomes or choices, one of which will be selected according to its assigned probability. These definitions result in the deterministic occurrence of events with respect to time and probabilistic occurrence of outcomes with respect to probability.

Analysis is provided by unravelling a choice-point net into an augmented reachability graph. The nodes and edges of this graph contain information about events, timing, probability and the location of objects within the system. Health-care professionals may then translate their questions into mathematical properties with respect to the information in the graph, i.e., events, time, probability and location. These translations are termed question restrictions because they restrict the set of paths in the graph which answer the question. Through the use of recursive algorithms and the given properties, it is then possible to ask questions about a protocol and receive answers. Preliminary research on choice-point nets, augmented reachability graphs and question restrictions has been published in [64], [65].

To improve computational feasibility, a more efficient approach to analysis is provided. Unravelling the complete behaviour of a choice-point net into an augmented
reachability graph is unfortunately subject to significant state-space explosion. A model of a health-care system and protocol is likely to possess too many unique states to feasibly produce a complete picture of its behaviour. Given a particular question, it is likely that only a small portion of the graph is relevant to the answer. Therefore, when a question is posed its associated set of restrictions is used to dynamically generate a restricted directed acyclic graph from a choice-point net. This graph consists solely of paths that satisfy the restrictions and answer the question, therefore reducing the computation required to find an answer.

The remainder of this thesis is organized as follows. Chapter 2 introduces the mathematical theory upon which this research is based. A review of relevant literature is provided in Chapter 3. Choice-point nets, their associated augmented reachability graphs, and the algorithms to convert the first into the second are defined in Chapter 4. Proof that these algorithms produce behaviourally equivalent data structures is provided in Chapter 5. Question restrictions, the means by which questions about a protocol are formed and answered, are described in Chapter 6. They are put to more efficient use in Chapter 7 where a reduced form of augmented reachability graph, known as a restricted directed acyclic graph, is introduced. The modelling approach described in this dissertation is then applied to a real-world protocol to control respiratory infection outbreaks in long-term care homes in Chapter 8 as a proof-of-concept. An abridged form of this model is analyzed in Chapter 9; three questions are posed and answered for three different strategies, then compared. Conclusions drawn from this research and suggested future work are then provided in Chapter 10.
Chapter 2

Background

The research contained in this dissertation depends upon, draws inspiration from, or is related to several fields of study. The fundamental concepts from these fields are presented here to provide the background necessary for future chapters.

2.1 Petri Nets

An ordinary Petri net \([50], [49]\) \(\mathcal{N} = (P, T, \zeta)\) is comprised of a set of places \(P\), a set of transitions \(T\) and an incidence relation \(\zeta : (P \times T) \cup (T \times P) \to \mathbb{N}\) that connects places to transitions via some number of input arcs and vice-versa via some number of output arcs, where \(\mathbb{N}\) is the set of natural numbers. These directed edges do not connect these objects to other members of their respective sets. The incidence relation \(\zeta\) is a multiset \([59]\) and therefore may contain identical entries, \(i.e.,\) multiple connections between a place and a transition are permitted.

Places and transitions are fundamentally different from a modelling perspective: places represent options while transitions represent actions that may be taken. A
transition $t \in T$ has two sets of directed edges: those incoming from the *preset* $\cdot t = \{ p \mid \zeta(p, t) > 0 \}$ and those outgoing from the *postset* $t^* = \{ p \mid \zeta(t, p) > 0 \}$.

$$
\begin{array}{c}
\begin{array}{c}
\text{Figure 2.1: An Example of an Ordinary Petri Net}
\end{array}
\end{array}
$$

The Petri net in Figure 2.1 has three places, represented by circles, and three transitions, represented by vertical bars. A directed edge from a place $p$ to a transition $t$ indicates that $\zeta(p, t) > 0$; an edge from a transition to a place implies $\zeta(t, p) > 0$.

The leftmost transition has a preset and postset of size one. The same can be said for the middle transition—even though it has two incoming arcs, they both originate at the same place. The rightmost transition, however, has a preset with one input place and a postset containing the two other places in the Petri net.

### 2.1.1 Marked Petri Nets

Within a Petri net, each place may contain any number of indistinguishable uniform objects known as *tokens*. As events occur within the net, tokens are transferred along the associated transition’s edges from its preset places to its postset places. A *marking function* $M : P \rightarrow \mathbb{N}$ is a function that maps a place to the number of tokens stored in that place. This mapping is also extended to vectors where $M : P^{\mid P\mid} \rightarrow \mathbb{N}^{\mid P\mid}$.

Here, the domain is a vector of places and the codomain contains numerical entries for the respective locations. A single marking is defined as $m \in \mathbb{N}^{\mid P\mid}$ and has the form $[ M(p_1) \ M(p_2) \ \ldots ]$. Note that the number of tokens for any given place can never be
negative. Figure 2.2 contains a sequence of transition firings for the Petri net shown in Figure 2.1. Tokens are represented by small, solid circles within the places.

Using the order of the places in the diagram from left to right, a triple $[x_1 x_2 x_3]$ can be used to describe a given marking where $x_1$ is the number of tokens in the leftmost place, $x_2$ is the number of tokens in the center place and $x_3$ is the number in the rightmost place. Such a vector containing the marking values for the net is referred to as a marking and is denoted by $m$. Figure 2.2 illustrates a firing sequence for the net shown in Figure 2.1. The initial marking is $m_0 = [2 0 0]$ (Figure 2.2(a)); after firing the leftmost transition once (Figure 2.2(b)) and again (Figure 2.2(c)), followed by the center transition (Figure 2.2(d)), the rightmost transition fires leading to marking $m = [1 1 0]$ (Figure 2.2(e)).

A marked Petri net $N_M = (P, T, \zeta, M, m_0)$ contains the same elements as an ordinary Petri net in addition to a marking function $M : P \rightarrow N$ and an initial marking $m_0$ for the net. A vector that contains the marking values for the net is denoted by $m$. The current marking for a Petri net is analogous to a current state in an FSA; the same can be said for initial markings and initial states. With respect to Petri nets, the terms marking and state will be used interchangeably.

If a marked Petri net $N_M$ contains only one token ($\exists p \in P, M(p) = 1$ and $\forall p' \in (P- \{p\}), M(p') = 0$) and all transitions have exactly one input and output place ($\forall t \in T, |\bullet t| = |\bullet t| = 1$) then it is equivalent in its operation to a finite-state automaton. An example of this situation is shown in Figure 2.3.

A marked Petri net $N_M$ is $k$-bounded if no place contains more than $k$ tokens at any reachable state. The net $N_M$ is considered bounded if it is $k$-bounded for some value of $k$. 
Figure 2.2: An Example of Sequential Firing of Transitions in an Ordinary Petri Net
2.1.2 Labelled Petri Nets

The Petri nets seen thus far operate “silently”, i.e., no language is generated when the net is active. This can be remedied by using a labelled Petri net $N_L = (P, T, \zeta, M, m_0, \Sigma, L)$. It has all the components of a marked Petri net with the addition of an alphabet $\Sigma$ and a labelling function $L : T \rightarrow \Sigma$. This mapping is easily extended to strings from the alphabet such that $L : T^* \rightarrow \Sigma^*$. Figure 2.4 contains the net shown in Figures 2.1 and 2.2 labelled with the alphabet $\Sigma = \{a, b, c\}$. The firing sequence illustrated in Figure 2.2 would generate the string $aabc$.

2.1.3 Reachability Graphs

A reachability graph (or graph of markings) represents the sequences of events and reachable states generated by the Petri net. Each node in the graph contains a marking which is possible for the net to achieve. The edges of the graph are each associated with a transition—a source node is connected to an end node if and only if
the edge’s transition is enabled in the source marking and produces the end marking once fired. Figure 2.5 contains the reachability graph for the labelled Petri net shown in Figure 2.4 with two tokens initially in the leftmost place.

![Figure 2.5: An Example of a Reachability Graph](image)

### 2.2 Discrete-Event Systems

A discrete-event system (DES) is a process whose operation is defined solely in terms of its actions. It is characterized by sequences of events where, at each stage of the process’s execution, what may occur depends on the events that have occurred up to that point. The control of discrete-event systems was introduced in [53]; the fundamentals of DES theory can be found in [54]; broad but comprehensive documentation is given in [16]; and [61] is an excellent introductory survey. One of the most common modelling mechanisms for discrete-event systems is the finite-state automaton (FSA).

The system under consideration is known as the plant. Its independent operation is typically characterized by a five-tuple

$$G = (\Sigma, Q, \delta, q_0, Q_m)$$

where $\Sigma$ is the alphabet of events, $Q$ is the set of states, $\delta$ is the transition function, $q_0$ is the initial (start) state and $Q_m$ is the set of marked (end) states. The function $\delta$ may
be totally or partially defined where \( \delta : \Sigma \times Q \rightarrow Q \). Note that if \( \delta \) is a partial function, states may not have transitions defined for each event. The definition for \( \delta \) is also extended recursively to handle strings; formally, \( \delta(\sigma_1\sigma_2\ldots\sigma_n, x) = \delta(\sigma_2\ldots\sigma_n, \delta(\sigma_1, x)) \) for some natural number \( n \geq 2 \).

The automaton \( G \) mimics the actions of the process through its associated languages; \( L(G) = \{ s \mid s \in \Sigma^* \land (\exists q \in Q)(\delta(s, q_0) = q) \} \) characterizes the system’s operation while \( L_m(G) = \{ s \mid s \in \Sigma^* \land (\exists q \in Q_m)(\delta(s, q_0) = q) \} \) represents recognized “completed” sequences. If \( G \) is a finite-state automaton then \( L(G) \) and \( L_m(G) \) are both regular languages.

An example of a discrete-event system plant can be found in Figure 2.6; it is identical to the reachability graph shown in Figure 2.5. Here, the alphabet is \( \Sigma = \{a, b, c\} \) and the state set is defined by \( Q = \{1, 2, 3, 4\} \). The initial state is \( q_0 = 1 \) and there is one marked state \( (Q_m = \{4\}) \).

### 2.2.1 Working With the Plant

Discrete-event system applications go far beyond modelling a given process. The purpose of research in this field is to provide the means to better understand and control these types of systems. These goals are achieved through the application of different *agents* to the system, all of which are characterized in the same manner as
the plant (e.g., finite-state automaton).

One such agent is a supervisor $S$ which observes and controls the operation of the plant by enabling or disabling specific events at specific states within $G$. However, there exist some events the supervisor may not be able to eliminate; some are controllable and some are uncontrollable. Formally, $\Sigma = \Sigma_c \cup \Sigma_{uc}$ and $\Sigma_c \cap \Sigma_{uc} = \emptyset$. Controllable events may be disabled by a system controller at any given state while uncontrollable events can never be cancelled. Under supervision, the plant can only generate strings which it is physically capable of producing and which are permitted by the supervisor. Formally, $L(S/G) = L(S) \cap L(G)$.

2.2.2 Using Petri Nets

Although FSAs are widely considered the data structure of choice in DESs, there exists a significant body of research dedicated to the use of Petri nets as the modelling formalism [43], [34]. With respect to supervision, the goals are identical: synthesize a supervisor that, when applied to the plant, enforces control and produces a desired result. However, Petri nets are typically more succinct yet more expressive than FSAs, and offer many established methods for analysis.

Supervision is typically applied via a set of control places that are connected to transitions by arcs. A control is a function that maps control places to $\{0, 1\}$. Transition enablement is determined by both a standard Petri net marking and a control. The goal of supervision is to ensure the net either occupies legal markings (and avoids forbidden ones), generates a legal language (and does not produce illegal strings) or avoids deadlock.
2.3 Mathematical Epidemiology

Traditionally, modelling in the field of health-care has been focused on examining the transmission of disease within a population. The first known attempt is a mathematical model created by Daniel Bernoulli in 1760 of the impact of inoculation on smallpox [14]. The introduction of the compartmental model, published in three papers between 1927 and 1933 by W. O. Kermack and A. G. McKendrick, provided the next significant leap forward in the field. Kermack and McKendrick assigned each individual to one of three classes or compartments: susceptible, infectious and removed. In this SIR (Susceptible, Infectious and Removed) model, people shift between the three classes according to the chart shown in Figure 2.7. As susceptible individuals mix with infectious individuals (with a transmission rate of $\beta$) and those who are infected recover (at a rate of $\gamma$), the content of each compartment changes. This is expressed as a system of differential equations:

$$\frac{dS}{dt} = -\beta SI$$
$$\frac{dI}{dt} = \beta SI - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

By calculating a numerical approximation of the solution, it is possible to determine $S(t)$ and $I(t)$, i.e., the number of susceptible and infectious individuals at time $t$, respectively. The evolution of $S$ is usually presented as a graph of $S$ as a function of...
Chapter 2. Background

$t$; the same can be said for $I$. Compartmental models are heavily influential in the field and have been augmented into more complex forms [68].

Other approaches to epidemic models include stochastic models that use discrete/continuous-time Markov chains or stochastic differential equations and network models that use graphs to represent contact networks [14]. While these comprise a vital component of health-care modelling, they are primarily concerned with the transmission of disease and the effectiveness of medical intervention (e.g., vaccination), not the protocols adopted by health-care organizations during an outbreak.

2.4 Disaster Response

In contrast to mathematical epidemiology, disaster response modelling is concerned with evaluating the ability of a physical system to manage an emergency. Topics in this field include routing relief supplies [7], stockpiling and distributing medical supplies [68] or determining optimal locations for disaster recovery centres [25]. However, much of the field is concerned with the medical consequences of disasters such as patient surges at hospitals. These disasters may not be not medical in origin (e.g., earthquakes [28]) but many of them are (e.g., influenza [30]).

The body of literature in this field is significant but the modelling techniques fall into one of two camps: simulation-based and non-simulation. Techniques that avoid simulation (e.g., stochastic inventory control [7], compartmental models [42]) may be less detailed and/or forced to assume some homogeneity with respect to behaviour and reaction. In contrast, simulation models can be extremely complex and diverse; some agent-based models assign individual properties to vast numbers of agents based upon national census data [30]. The pitfall is that the model is so complex that simulation
is the only analysis available while non-simulation models provide more formal means of investigating the system.

Both approaches to disaster modelling provide estimates, forecasts or results relating to variables of interest within the system. Single values or point estimates are by far the norm offered for predictions. Common estimates include number of deaths, costs per person, bed occupancy and patient processing times. Sensitivity analysis is often performed to determine how “sensitive” the results are to changes to the inputs, i.e., how results vary if the input values are incorrect or altered. There is no locatable discussion pertaining to human error, nor to determining several possible outcomes and the probabilities thereof.
Chapter 3

Literature Review

Initially, the purpose of this research was to locate an appropriate existing discrete-event modelling framework for health-care protocols and devise additional methods of analysis specific to the problem. There are several requirements for a suitable modelling structure:

- it must incorporate timed events
- it must incorporate probability on event outcomes
- it must be scalable
- it must be somewhat intuitive
- it must provide useful forms of analysis

However, a lengthy search yielded nothing that was truly appropriate.

Finite-state automata, which are the original structure of choice for discrete-event systems [53], [16], are not scalable. Given the large number of people and resources that must be modelled, FSAs are subject to overwhelming state-space explosion.
Individual elements would each be modelled using an FSA and the system as a whole would be constructed by shuffling [16] all of the automata into a single, massive structure. The state-space explosion is unfortunate, as the analysis from such a model would be extremely specific: each state in the combined system would contain detailed state information about each individual element.

Limited lookahead control [20], [66] offers a novel solution to the problem of an unmanageable state space. Rather than calculate a complete supervisor offline, which may not be feasible, an $N$-step “lookahead” tree of possible system behaviour is analyzed to determine control policy. While this approach mitigates state-space explosion it also limits analysis of the system to a window of arbitrary size. Parameterized DESs [10] considers a system composed of $N$ similar processes and creates a scalable control policy by exploiting the replicated nature of the plant. This approach is successful at reducing state-space explosion from a supervisory perspective, but not from a combined behaviour perspective.

Stochastic [62] and probabilistic [45], [51] DESs assign probabilities to events depending on the current state. Timed DESs [13], [33] associate time with events, giving them lower and upper bounds on time to occurrence. While each of these satisfies a requirement, it also lacks the requirement met by the other and both suffer from the potential of insurmountable state-space explosion.

To satisfy probability and time requirements, one modelling option would be Markov chains [16], [18]. These offer states (i.e., possible values of a random variable), one of which is inhabited by the system at index $n$, which usually represents time. Probability is associated with transitioning from one state to another. Markov decision processes [63], [35] offer an extension with probabilities associated with actions
that change state, rewards associated with actions and policies for selecting actions. They can be discrete-time (non-negative integers) or continuous-time. Unfortunately, in both Markov chains and Markov decision processes the indexing mechanism for time precludes events occurring at unevenly-spaced preset times. In addition, both these modelling formalisms may be subject to insurmountable state-space explosion.

The significant issue of overwhelming state-space explosion renders the aforementioned promising approaches unsuitable. Tracking the individual states of each resource or person is not feasible. In contrast, Petri nets [50], [49] can be scaled with less impact thanks to indistinguishable tokens: numbers can be easily increased or decreased without affecting the system’s structure and the system state is only dependent on how many resources or people are in specific states, not which ones are in them.

Stochastic Petri nets (SPNs) [48], [5] offer transitions timed with rates and calculations to determine the probability of an enabled transition firing first in a given marking. Generalized stochastic Petri nets (GSPNs) [19], [49] [17] go further by introducing weighted immediate transitions and calculations to determine the probability of an enabled transition firing first in a given marking based on rate or weight. Unfortunately, the SPN and GSPN approach to timing (rates on transitions and random firing delays generated with exponential distributions) is ultimately unsuitable due to the system behaviour that must be accommodated. Institutions of interest to emergency protocols, such as hospitals, operate under strict schedules that result in a deterministic occurrence of events. For example, meals are served to patients three times per day. If modelled using a timed transition with a rate of three, the resulting representation would assume a mean time to occur of eight hours—patients
would eat around 8:00 a.m., 4:00 p.m. and midnight. The ability to deterministically time events, define different outcomes and assign a probability to those outcomes is therefore essential.

Timed Petri nets [24], [3] associate time with either places (*P-timed*) or transitions (*T-timed*). In the former, tokens are unavailable (*i.e.*, can not be used to fire a transition) for a period of time after being deposited in a place. In the latter, a transition reserves tokens (*i.e.*, does not permit other transitions to use them) for a period of time before firing. In contrast, time Petri nets [9], [31] assign each transition a time interval and a clock tracking the time since it was first enabled. The time on this clock must be within the interval for the transition to fire. Neither of these approaches provide a means to attach probability to events and/or more than one option for the timing of events.

Coloured Petri nets [40], [29] augment Petri nets with a programming language. Data values (or “colours”) are attached to tokens and places may only contain tokens of certain colours. Arc expressions, which operate like a programming language, define which tokens are fired by a transition and alter the data values on those tokens. Transitions can be assigned Boolean expressions (*guards*) which must evaluate to *true* for enablement. Time may be integrated using a global clock and tokens carrying timestamps. The variety of analysis available is impressive (*e.g.*, simulation, verification) but some approaches are subject to state-space explosion. In addition, probability is only introduced in simulation and the approach to timing is not flexible enough for our application to health-care protocols.

Finally, DEVS (Discrete Event System Specification) [70], [69] shares a similar name with DES, but is different. Both are based upon states and discrete events,
but DEVS is a modelling approach which is primarily concerned with simulation. States are defined sequentially and events are considered inputs to or outputs from the system. There are two transition functions: one internal (based only upon the current state) and one external (based upon the current state and an input event). Output is generated based upon the current state. This simulation-based approach with inputs and outputs does not time events or assign probabilities to different outcomes.

A summary of this review of discrete-event modelling frameworks can be found in Tables 3.1 and 3.2.

### 3.1 Modelling Health-Care Response to a Disaster

The body of literature related to modelling and analyzing a system’s response to a disaster is extensive; [12] lists a selection of over 80 references containing such models, most of which are concerned with the performance of a health-care system. Some indicative selections from this list are described in this section, along with an additional relevant approach that was located.

The ability to respond to bioterrorism such as an anthrax attack is one such disaster that has been studied. Discrete-event simulation is used by [36] to model antibiotic distribution centers and suggest optimal staffing levels. A compartmental model undergoes simulation in [11] to compare the cost of approaches to reduce mortality (e.g., universal vaccination and emergency surveillance and response). Pandemic influenza is also a concern; spreadsheet-based software known as FluSurge has been used in
<table>
<thead>
<tr>
<th>Structure</th>
<th>Timing</th>
<th>Probability</th>
<th>Issues Regarding Application</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited Lookahead DES</td>
<td>-</td>
<td>-</td>
<td>Mitigates state-space explosion but hinders analysis</td>
<td>[20], [66]</td>
</tr>
<tr>
<td>Parameterized DES</td>
<td>-</td>
<td>-</td>
<td>Mitigates state-space explosion from a supervisory perspective</td>
<td>[10]</td>
</tr>
<tr>
<td>Timed DES</td>
<td>Assigns timing intervals to events</td>
<td>-</td>
<td>No approach to probability; subject to state-space explosion</td>
<td>[13], [33]</td>
</tr>
<tr>
<td>Probabilistic / Stochastic DES</td>
<td>-</td>
<td>Assigns probabilities to events</td>
<td>No approach to timing; subject to state-space explosion</td>
<td>[45], [62], [51]</td>
</tr>
<tr>
<td>DEVS</td>
<td>Governs internal transitions with a time advance function</td>
<td>-</td>
<td>No approach to probability; simulation-based analysis</td>
<td>[70], [69]</td>
</tr>
<tr>
<td>Markov chains</td>
<td>Typically used as the index for the system’s state</td>
<td>Defined over changes of state</td>
<td>Approach to timing is not appropriate</td>
<td>[16], [18]</td>
</tr>
<tr>
<td>Markov decision processes</td>
<td>Discrete or continuous decision epochs</td>
<td>Defined over actions that change the current state</td>
<td>Approach to timing is not appropriate</td>
<td>[63], [35]</td>
</tr>
<tr>
<td>Markov nets</td>
<td>-</td>
<td>Assigned over place post-sets with at least two transitions</td>
<td>No approach to timing</td>
<td>[8]</td>
</tr>
</tbody>
</table>

Table 3.1: Summary of Discrete-Event Modelling Approaches (Part 1)
<table>
<thead>
<tr>
<th>Structure</th>
<th>Timing</th>
<th>Probability</th>
<th>Issues Regarding Application</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Petri nets</td>
<td>Intervals associated with transitions</td>
<td>-</td>
<td>No approach to probability</td>
<td>[9], [31]</td>
</tr>
<tr>
<td>Timed Petri nets</td>
<td>Delays associated with places or transitions</td>
<td>-</td>
<td>No approach to probability</td>
<td>[24], [3]</td>
</tr>
<tr>
<td>Stochastic Petri nets</td>
<td>Rates on transitions used to generate random firing delays</td>
<td>Calculated using transition rates</td>
<td>Approach to timing and probability is not appropriate</td>
<td>[48], [5]</td>
</tr>
<tr>
<td>Generalized stochastic Petri nets</td>
<td>Rates on transitions used to generate random firing delays</td>
<td>Calculated using transition rates and weights</td>
<td>Approach to timing and probability is not appropriate</td>
<td>[19], [49], [17]</td>
</tr>
<tr>
<td>Coloured Petri nets</td>
<td>Attached to tokens to introduce firing delays</td>
<td>In simulation</td>
<td>Probability only available in simulation</td>
<td>[40], [29]</td>
</tr>
</tbody>
</table>

Table 3.2: Summary of Discrete-Event Modelling Approaches (Part 2)
to estimate the resources required such as ICU and non-ICU hospital beds as well as ventilators. An agent-based model known as PHIMs (Public Health Interactive Model & simulation) is described in [52] as an interactive simulation environment which allows users to make policy decisions regarding the management of pertussis (whooping cough) and examine their impact.

Many of the proposed modelling approaches are not disaster-specific. Although [37] does use an airport accident as an example (comparing approaches to prioritizing patients for transport to hospital), its simulation approach implemented in Visual Basic.net is not limited to that emergency situation. No specific disaster is discussed in [41]; the concern is any emergency which produces a surge in pediatric patients in New York City. Simulation performed in a spreadsheet compares three management strategies (regional distribution of patients, limiting care to essential interventions and the combination of the two) and estimates the number of patients in ICU and non-ICU beds and deaths.

3.2 Petri Nets and Health-Care Modelling

The application of Petri net theory to the problem of health-care modelling is not common; only five instances of research on the topic could be located. In [39], the business process redesign methodology is applied to the problem of intake procedure for new patients requesting non-urgent care at a mental health institution. These design practices are typically associated with commercial applications, but they were employed here for a redesign of the intake system. Both the original and new intake procedures were modelled as a coloured Petri net simulation, which generated results (average, low and high) for both flow time and service time.
A similar theme plays out in [6] which combines business process modelling and Petri net theory with the purpose of improving large-scale software system design. The chosen process is the Design & Engineering Methodology for Organizations (DEMO), which “studies communication, information, and action within the context of an organization”. The system is then graphically represented using Petri nets. Of interest is the example where a medical center’s patient admission system is modelled using this approach and analyzed using existing Petri net tools for structural issues such as deadlock and conflict.

The goal of [4] is to model the spread of directly transmitted infectious diseases using coloured stochastic Petri nets (CSPNs). Vertical transmission (from mother to child) is modelled in addition to transmission resulting from mixing (person-to-person contact). An SI model (Susceptible and Infectious) is produced, which is similar to the classic SIR but without the “removed” class, implying that the infection does not result in immunity or death. Simulation is used to predict values such as the probability of disease extinction.

Emergency medical services are modelled in [67] using two cases: all patients have equivalent priority and patients may have different priorities. The former model is found to be bounded and free from deadlock. Emergency department overcrowding is the topic of [23], which models some hospital operations as Petri nets and analyzes them to determine how the system moves from an initial acceptable marking to an unacceptable marking. This is accomplished via mixed-integer programming using vector and matrix elements constructed from the net.
3.3 Further Related Health-Care Research

Not all of the literature relevant to this thesis can be easily categorized. This section describes several approaches of interest which are somewhat disparate yet pertinent to the research documented in this dissertation.

Improving processes in hospitals has been the subject of some recent research. Efficient operating room (OR) scheduling is the goal in [2] where each operation is broken into five stages: patient transport to the OR (potentially waiting until an OR is available); surgery; patient transport to the recovery area (potentially waiting until a recovery bed is available); cleaning of the OR; patient transport to a ward. The two scheduling strategies compared allow patients to recover in the OR if no recovery bed is available, or not. A mathematical model of the problem is created, Lagrangian relaxation is applied to some of the constraints and the model is solved using dynamic programming. The solution generated is described as infeasible; an algorithm is then used to generate a feasible solution.

Improving the pharmacy delivery system for “mobile medicine closets” in hospitals is the subject of [1]. Each hospital has a central pharmacy and one medicine closet per medical unit. Every week these closets must be transported to the central pharmacy (by foot, tractor or truck) for inventory and refill, then transported back to their respective units; [1] addresses this transportation problem. A mixed-integer linear programming approach was used to plan transport routes. The proposed solution was tested with discrete-event simulation with good results.

Current modelling techniques concerned with accident prevention may work well
for physical components, but [46] is concerned with applying this approach to “socio-
technical modeling” which takes social and organizational factors into account. Hu-
man error should be considered, including deviations from normal or effective proce-
dure and differences between specified and established practice. Safety defences are
often removed over time for cost-cutting and productivity, but decisions that seem
safe and insignificant when made alone may result in an unsafe combination. The
proposed approach, STAMP (Systems-Theoretic Accident Modelling and Processes)
and STPA (STAMP-based Analysis), has two goals: to identify hazards and safety
constraints and to eliminate or reduce violations of these constraints. Of particular
note is an examination of the E. coli water contamination in Walkerton, Ontario,
Canada in 2000. The water system safety control structure is modelled and analyzed,
the results of which predict or foreshadow an accident.

Discrete-event systems were employed in [15] to model public health interventions
on outbreaks of disease. A dynamic approach is employed where each infected indi-
vidual is represented by a module and the system’s state is represented by a vector of
state parameters for individuals. Supervision activates and alters intervention policies
made by a public health unit, which uses contact tracing to identify exposed indi-
viduals. In the model, symptomatic individuals are isolated and medicated once an
outbreak is detected while the public health unit contact traces and monitors those
exposed. Simulation was performed in MatLab with time steps for event occurrences.
From these results, the expected number of infected individuals and the variance on
that value were estimated.
Chapter 4

Choice-Point Nets and Augmented Reachability Graphs

The modelling of health care protocols requires the selection of a data structure that can accurately capture the schedules of the institution and the protocol itself. In addition, the chosen modelling technique must provide a method to represent different outcomes arising from the same event. This is necessary as the people responsible for implementing the protocol will have to choose between different actions and the probability of those decisions will depend on the circumstances at the time.

A choice-point net (CN) $\mathcal{N}_{CP} = (P, T, I, C, R, SL, ML, S, MD, CU, M, m_0)$ is a marked Petri net that has been given additional timing and probability information. Choice-point nets contain the following elements: a set of places $P$; transitions $T$; input arcs $I$; choices $C$; three timing functions $R$, $SL$ and $ML$; firing functions $S$ and $MD$; the number of clock units $CU$; marking function $M$ and initial marking $m_0$. In CNs the transitions $T$ are all timed, either using timers ($T_R$), a single lapse ($T_{SL}$) or multiple lapses ($T_{ML}$). Formally, $T = T_R \cup T_{SL} \cup T_{ML}$ where $\cup$ represents disjoint
union. Timing is defined and enforced with respect to \( CU \in \mathbb{N} \) which represents the number of ticks on a global clock and takes natural number values (e.g., a 24-hour clock would have \( CU = 24 \) and possible clock values of 0-23).

The firing of a transition is only permitted if its associated timer value is equivalent to that on the global clock or (at least one of) its associated lapse(s) has “counted down” to zero after it was most recently enabled. The timing function \( R \) maps associated transitions to clock values at which the transition may fire. The single lapse function \( SL \) maps transitions in \( T_{SL} \) to a single countdown value. In contrast, the multiple lapse function \( ML \) associates transitions with a multiset of lapses, each initialized upon the deposit of new tokens within the transition’s input places using the base value function \( B_{ML} : T_{ML} \to \mathbb{N} \). For example, consider a multiple lapse transition \( t \) where \( B_{ML}(t) = 24 \) and \( ML(t) = \{6, 18, 24\} \). If new tokens were deposited into \( t \)’s input places, its set of lapses would be updated to \( ML(t) = \{6, 18, 24, 24\} \).

Formally, \( R : T_R \to 2^\mathbb{N} \), \( SL : T_{SL} \to \mathbb{N} \) and \( ML : T_{ML} \to \mathbb{N}^{MS} \). Given a marking \( m \), current clock value \( 0 \leq d < CU \), future clock value \( 0 \leq e < CU \) and time difference \( f = (e - d + 24) \mod 24 \) where \( d, e, f \in \mathbb{N} \), a transition \( t \in T \) is able to fire at time \( e \) if

\[
(\forall p \in \bullet, M(p) \geq \zeta^I(p, t)) \land \quad ((t \in T_{SL}) \land (SL(t) = f)) \lor
\]
\[
((t \in T_{ML}) \land (f \in ML(t))) \lor
\]
\[
((t \in T_R) \land (e \in R(t)))
\]

where \( \zeta^I : P \times T \to \mathbb{N} \) is an incidence function that maps a place and transition to the number of input arcs from the given place to the given transition.

A helper function \( R_N : (T_R \times \mathbb{N}) \to \mathbb{N} \) is defined to determine the next possible time at which a timer transition may fire given the global clock’s present value. Formally,
for all transitions $t \in T_R$ and $d \in \mathbb{N}$ where $0 \leq d < CU$, 

$$R_N(t, d) = \begin{cases} \min \{e \mid e \in R(t) \land e > d\} & \text{if } \min \{e \mid e \in R(t) \land e > d\} \neq \emptyset \\ \min \{e \mid e \in R(t)\} & \text{otherwise} \end{cases}$$

Informally, the function selects the smallest clock value from $R(t)$ that is larger than $d$; if no larger value exists, the smallest clock value in the set is selected.

In choice-point nets, an enabling set for a given transition contains one token for each input arc where the token resides in the arc’s input place. A transition may possess one or many enabling sets depending on the marking. Tokens are interchangeable in these sets as they are indistinguishable from each other. The enabling degree function $ED : T \times \mathbb{N}^{[P]} \rightarrow \mathbb{N}$ determines the number of enabling sets for a given transition and a given marking. Formally, for all transitions $t \in T$ and markings $m \in \mathbb{N}^{[P]}$

$$ED(t, m) = \min \{ \left\lfloor \frac{M(p)}{C(p, t)} \right\rfloor \mid p \in \cdot \}$$

The manner in which each transition in $\mathcal{N}_{CP}$ may fire is dependent on its firing semantics. A transition in a standard Petri net will only use one token per input arc when it fires, no matter how many tokens may be present in its input places [50], [16]. This strategy is defined in choice-point nets as single-set semantics where only one enabling set is processed per firing. However, a transition with marking-dependent semantics may fire several sets of tokens at the same time in near-simultaneous succession. The number of times a transition may fire in this manner is determined by a marking-dependent function denoted by $md$, $md \in SFM$ where $SFM$ is the space of
all functions from markings to integers; formally, \( md : N^{|P|} \to Z \). Although these functions may produce any integer value, the net naturally restricts the number of times any transition \( t \in T_{MD} \) may fire in any marking \( m \) to within \([0, ED(t, m)]\). If the result of the marking-dependent function is negative, the net will not allow the transition to fire; if the result exceeds the enabling degree of the associated transition, the net will permit the transition to fire according to the enabling degree. A transition with multiple lapse timing must have \textit{multiple lapse-dependent} firing semantics based upon its current transition clock values. These transitions may fire once for each individual lapse in the clock set that has finished counting down. The set of transitions \( T \) is partitioned according to these firing semantics. Formally, \( T = T_S \cup T_{MD} \cup T_{MLD} \) where \( S : T_S \to 1 \) for single-set transitions \( T_S \), \( MD : T_{MD} \to SFM \) for marking-dependent transitions \( T_{MD} \) and \( MLD : 2^N \to N \) for multiple-lapse transitions \( T_{MLD} \). Since transitions with multiple lapse timing must have multiple lapse semantics, \( T_{ML} = T_{MLD} \).

Let \( \rho : T \to N \) represent the \textit{firing number function} which maps transitions to the number of times they may fire in near-simultaneous succession in marking \( m \) at time \( 0 \leq d < CU \) where \( d \in N \). These values are calculated using the aforementioned firing semantics for single-set, marking-dependent and multiple-lapse transitions. A transition \( t \in T \) may fire repeatedly at time \( d \) if

\[
\rho(t) > 0
\]

In choice-point nets, it is assumed that transitions with multiple lapses do not share input places with any other transition that could remove tokens before their associated lapses have counted down. However, single lapse and timer transitions are permitted to share input places. If several transitions may fire at time \( d \), possibly repeatedly,
it is possible that the transitions which fire first “steal” the tokens necessary to fire the remaining transitions. This will result in some transitions firing less often or even being pre-empted altogether. Such an approach reflects real-life conditions where the occurrence of one event cancels the occurrence of another. Formally, given a marking \( m \) and clock value \( 0 \leq d < CU \) where \( d \in \mathbb{N} \), transitions \( \{t_1, t_2, \ldots, t_n\} \subseteq T \) are in pre-emptive conflict at time \( d \) if they are all able to fire and

\[
\exists p \in (\bullet t_1 \cap \bullet t_2 \cap \ldots \cap \bullet t_n), (\rho(t_1) \times \zeta^I(p, t_1)) + (\rho(t_2) \times \zeta^I(p, t_2)) + \ldots + (\rho(t_n) \times \zeta^I(p, t_n)) > M(p)
\]

It is also possible that the firing of one transition may not affect the firing of another. Formally, given a marking \( m \) and clock value \( 0 \leq d < CU \) where \( d \in \mathbb{N} \), transitions \( \{t_1, t_2, \ldots, t_n\} \subseteq T \) are in a race at time \( d \) if they are all able to fire and

\[
\forall p \in (\bullet t_1 \cap \bullet t_2 \cap \ldots \cap \bullet t_n), (\rho(t_1) \times \zeta^I(p, t_1)) + (\rho(t_2) \times \zeta^I(p, t_2)) + \ldots + (\rho(t_n) \times \zeta^I(p, t_n)) \leq M(p)
\]

Depending on the set of transitions that may fire under current conditions in the net, some may fire repeatedly while some subsets may be in pre-emptive conflict or be in a race. All three of these scenarios are possible within a given firing set. To simplify matters, the term conflict will be used to informally describe situations where at least one of these scenarios is playing out.

In Petri nets, the firing of transitions is governed entirely by whether there are enough tokens in a transition’s input place for firing to occur. In choice-point nets, the firing of transitions is also governed by timing values and firing semantics. Each transition’s clock dictates when a transition may fire; each transition’s firing semantics dictate how often a transition may fire in near-simultaneous succession. Although
transitions with multiple lapse timing must have multiple lapse-dependent firing (and vice-versa), the other varieties of semantics may be “mixed-and-matched”:

<table>
<thead>
<tr>
<th>Firing</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marking-Dependent</td>
<td>Timer</td>
</tr>
<tr>
<td>Single-Set</td>
<td>Single Lapse</td>
</tr>
<tr>
<td>Multiple Lapse-Dependent</td>
<td>Multiple Lapse</td>
</tr>
</tbody>
</table>

The function $C_T : T \rightarrow C$ maps each transition to a set of output “choices” $C \subseteq 2^{\Sigma^* \times [0,1] \times 2^P}$ representing possible outcomes. Each choice $(l, b, O)$ contains a textual label $l \in \Sigma^*$, a probability $b \in [0,1]$ and a set of output places $O \subseteq P$. The firing of a transition $t \in T$ with a current marking $m$ forces the selection of a choice $c \in C$ to determine the flow of output tokens.

The definition of a CN ensures that each enabled transition will fire when its associated timer or lapse(s) allow(s) it to. Although this definition does not allow the possibility that an event may not occur, that scenario may still be modeled using choices. By defining a choice whose output arcs mirror the transition’s input arcs and assigning the probability that the event never occurs, it is possible to simulate the transition firing between zero and the maximum number of times permitted by the model.

Petri nets are in fact a special case of choice-point nets with certain timing and firing values: one may set $CU = 1$, make every transition a single lapse transition with a countdown of 1, assign single-set firing semantics to all transitions and ensure all transitions have a single choice with probability 1.0.

The Petri net notation $m|t|m'$ (i.e., the firing of transition $t$ in marking $m$ leads to marking $m'$) is insufficient for CNs; a single marking does not represent the true
state of $N_{CP}$ and a single transition no longer represents an event occurrence. The set of states of a choice-point net is denoted by $\Psi$. A single state $\psi \in \Psi$ where $\psi = (m, \chi, \rho, \theta)$ is comprised of four elements: a marking $m$, a transition clock function $\chi : T_{SL} \cup T_{R} \rightarrow \mathbb{N}$ or $\chi : T_{ML} \rightarrow 2^{\mathbb{N}}$ (depending on the context) which tracks values (countdown or lapse) for each transition, a firing number function $\rho : T \rightarrow \mathbb{N}$ which tracks the number of times a transition may fire and a global clock value $\theta \in [0, CU)$. These four values are needed to determine which transition(s) may fire next according to the timing constraints of the net.

The event set (or alphabet) $\Omega$ of a choice-point net is defined as $\Omega \subseteq T \times C \times [0,1] \times \mathbb{N}$. A single event $\omega \in \Omega$ where $\omega = (t, c, p, k)$ is composed of four elements: a transition $t \in T$, a choice $c \in C$, a probability $p \in [0, 1]$ and the number of clock ticks that must elapse before this event’s occurrence $k \in \mathbb{N}$. Thus, the notation $m|t) m'$ is replaced by $\psi|\omega) \psi'$ for choice-point nets or, more explicitly, $(m, \chi, \rho, \theta)|t, c, p, k)(m', \chi', \rho', \theta')$. Some of the elements in event $\omega$ are redundant, i.e., they can be determined from $\psi$ and $\psi'$. For example, $k = \theta' - \theta$. However, they are included in the definition of $\Omega$ to enhance the “readability” of the firing, i.e., to make it clear exactly what altered the state of the system and how.

The language $L(N_{CP}) \subseteq \Omega^*$ of the choice-point net $N_{CP}$ consists of all sequences of transitions and choices that may occur under the timing and probability constraints of the net.

### 4.1 A Small Example of a CN

Consider a long-term care home (LTCH) for elderly people. Each resident is seen by a staff member once in the morning and once in the night: 8:00 AM and 11:00
PM, specifically. While everyone in the facility received their influenza shots some months ago, two new residents who are infected but initially asymptomatic have since arrived. Health care providers are trained to look for a wide variety of symptoms, including those associated with colds and influenza. Should symptoms of that nature be identified, the resident must be isolated for 24 hours to limit exposure amongst the population and to ensure more serious symptoms do not develop. A CN model of the system is shown in Figure 4.1.

![Figure 4.1: Choice-Point Net Representation of a Long-Term Care Home Protocol](image)

The system is in initial marking $m_0 = [2200] = [M(s_{Normal})M(r_{Ill})M(r_{Isolated})M(r_{InPopulation})]$. The tokens that represent residents will move between $r_{Ill}$ for those who are unwell, $r_{Isolated}$ for those in isolation and $r_{InPopulation}$ for those who are allowed to move freely the LTCH. The tokens representing staff members do not move out of $s_{Normal}$ because they have been immunized against this particular strain. Since the protocol is defined in terms of days, the number of clock units has been set to 24, each one representing an hour. Visits by health care providers are represented by the transition $s_{SeeIllResident}$ which is associated with timers 8 and 23 that represent 8:00 AM and 11:00 PM on a 24-hour clock. Since visits from staff members will occur in parallel, $s_{SeeIllResident}$ has
marking-dependent semantics (denoted in the diagram by “MD”): for all markings \( m \), \( md(m) = ED(s\text{SeeIllResident}, m) \). Two different choices are assigned for this transition: the choice (Ignore, 0.2, \( O_{Ign} \)) with dashed lines represents the 20% likelihood that a staff member will not observe a resident’s symptoms; the choice (Isolate, 0.8, \( O_{Iso} \)) with dotted lines represents the 80% chance that a staff member will recognize that a resident is ill and isolate him or her. Since each resident is required to stay in isolation for 24 hours, transition \( r\text{DoneIsolation} \) has multiple lapse semantics with a base value of \( B_{ML}(r\text{DoneIsolation}) = 24 \) clock ticks which must elapse before the transition can transfer a token from \( r\text{Isolated} \) to \( r\text{InPopulation} \). This reflects the requirement that each resident must stay in isolation for 24 hours after he or she has been discovered to be ill.

It is important to note that the net shown in Figure 4.1 departs slightly from the standard manner in which Petri nets are drawn. Specifically, transitions have both incoming and outgoing arcs drawn on all sides. In formal Petri net models incoming arcs are drawn on one vertical side of a transition and outgoing arcs on the other. The departure here is designed to reduce arc crossings.

## 4.2 Augmented Reachability Graphs

To capture system behaviour so that choice-point nets can be analyzed, a net may be deterministically *unravelled* into a directed graph having timing and probability information. Formally, an *augmented reachability graph* \( RG_{CP} = (\Omega, X, \delta, x_0) \) contains an alphabet (identical to that of \( N_{CP} \)), a set of nodes \( X \), a transition function \( \delta \), and an initial node \( x_0 \).

The nodes and edges in the augmented reachability graph (ARG) are specialized
to contain additional information about the net moving from state to state. Each node \((m, \chi, \rho, \theta) \in X\) is comprised of a marking \(m\), a transition clock function \(\chi : T_{SL} \cup T_R \to \mathbb{N}\) or \(\chi : T_{ML} \to 2^\mathbb{N}\) that, depending on the context, maps each transition to its next timer or countdown(s) of ticks, a firing number function \(\rho : T \to \mathbb{N}\) that maps transitions to the number of times they may fire “simultaneously” and the value of the global clock \(\theta \in \mathbb{N}\). The purpose of \(\rho\) is to simulate the parallel occurrence of events by firing the transition repeatedly and in immediate succession.

Each edge corresponds to the firing of a transition and the selection of a choice. Formally, an edge is represented by \((t, c, q, k) \in \Omega\) where \(t\) is the transition that has fired, \(c\) is choice that was selected, \(q\) is the probability that the choice would be made, and \(k\) represents the time that has elapsed since the occurrence of the last event. In the event of a conflict, edges associated with transitions that did not fire first have \(k = 0\). For each transition \(t \in T\), \(C_T(t) = \{c_1, \ldots, c_n\}\) represents \(t\)’s choices with \(c_i = (l_i, b_i, O_i)\) and one must have \(\sum_{i=1}^{n} b_i = 1\).

The language of the augmented reachability graph \(\mathcal{RG}_{CP}\) is denoted by \(L(\mathcal{RG}_{CP}) \subseteq \Omega^*\). This language represents all possible sequences of events that may occur within the graph, which will be proven to be equivalent to the language of the associated choice-point net. Formally, \(L(\mathcal{RG}_{CP}) = L(\mathcal{N}_{CP})\).

The number of states within an augmented reachability graph depends on the number of possible markings, timing, firing and clock values that can be generated by the net. If the net can produce an infinite number of markings, then the associated graph will also contain an infinite number of states.
4.3 A Small Example of an ARG

Suppose that the management of the LTCH in the previous example wants to evaluate the effectiveness of its isolation protocol. For example, what is the probability that both residents’ influenza will be reported 24 hours after they began to show symptoms? How likely is it that cold and influenza symptoms will not be observed by health care providers more than once? To perform this type of analysis, the system model shown in Figure 4.1 has been deterministically unravelled into the ARG illustrated in Figure 4.2.

Each node shown in the display consists of six elements: first, a marking from the choice-point net; second, the timer value for transition $sSIR$ (abbreviated to $sSIR$ in subscript with alarm and marking-dependant in superscript); third, the lapse time “counting down” for $rDI$ ($rDI$ in subscript with multiple-lapse in superscript); fourth and fifth, the firing values of $\rho_{sSIR}$ and $\rho_{rDI}$ which track the number of times transitions $sSIR$ and $rDI$ may fire in near-simultaneous succession, respectively; sixth, the value of the global clock after a transition has fired. The firing values are essential to distinguish between nodes that may possess identical marking and timing values but represent different stages within a firing. Consider a sequence where a single transition $t$ that does not alter the marking may fire six times in near-simultaneous succession. Most of the nodes in that sequence would appear identical to each other if not for the different values of $\rho(t)$.

The ordering of the marking is $[rIll \ rIsolated \ rInPopulation]$. The marking does not include an entry for the number of staff members because that value stays constant; no transitions in the example change the number of tokens in $sNormal$. The
Figure 4.2: Augmented Reachability Graph of the CN Model for the LTCH Protocol edges are labeled with the names of the transition and choice, the probability of that choice and the time that has passed between the firing of transitions. Events that do
not have different outcomes, such as \texttt{rDonelIsolation}, are given a single choice with a probability of one.

There are two instances in the graph where \texttt{sSeeIllResident} and \texttt{rDonelIsolation} are able to fire at the same time, resulting in a race and the exploration of all possible sequences of choices. To illustrate these races, the edges are drawn using dotted lines. It is assumed that events in conflict occur in near-instantaneous succession. The time elapsed since the firing of the last transition is given to the first edge in a conflict path; all other edges in the same path are given an elapsed time of zero.

When resolving a conflict, a number of transitions may fire, each of which will have any number of associated choices. The probabilities of all of these choices are renormalized to reflect the number of different options. Let $CON = \{t_1, \ldots, t_n\}$ be the set of transitions that may fire and $C_T(t_i) = \{c_{i1}, \ldots, c_{im_i}\}$ be the set of choices for transition $t_i$. To normalize the probability of all choices $c_{ij}$ for all transitions $t_i$, $b$ must be divided by:

$$b_{CON} = \sum_{i=1}^{n} \sum_{j=1}^{m_i} b_{ij}$$

$$= \sum_{i=1}^{n} 1 \quad \text{(Assumption that each choice’s probabilities sum to one)}$$

$$= n$$

It is assumed that any transition in the conflict set may fire first and that no transition is more likely than any other to do so. The probabilities of the associated choices must be adjusted to enforce this rule. For example, when events \texttt{sSeeIllResident} and \texttt{rDonelIsolation} are in a race the probabilities of the three possible choices (Isolate, 0.8), (Ignore, 0.2) and (Done, 1.0) are each divided by $(0.8 + 0.2 + 1.0) = 2.0$, as
shown in Figure 4.2.

# 4.4 ARG Construction

This section is dedicated to the definition of five algorithms that will construct an augmented reachability graph from a given choice-point net. An augmented reachability graph for a CN contains all possible firings of the net under the given timing constraints. When calculating these firings, it is essential to maintain correct values for each transition’s timer, single or multiple lapse clock; Algorithm 1 is given this task. This function must be called every time a sequence of transitions completes firing to update each transition’s clock. If a transition has just been disabled, its clock must be disabled; if a timer or single lapse transition is newly-enabled, its clock must be reset; if a multiple lapse transition had new enabling sets deposited in its input places, new lapses must be added.

Once the transitions clocks are updated, the next step is to do the same for the firing numbers; Algorithm 2 is responsible for this task. Once the set of transitions whose clock values permit them to fire next has been determined, the firing numbers are updated according to transition timing semantics: marking-dependent transitions use the result of their function applied to the current marking; multiple lapse transitions are given the number of identical minimum lapses in their clock set; and single lapse transitions (which by definition fire once) are provided with a value of one.

Algorithms 1 and 2 are responsible for adjusting elements within a node, but it is Algorithm 3 that creates each new node. This algorithm essentially copies the elements of its parent and adjusts them to reflect the firing of a particular transition and the selection of a particular choice. It begins by testing whether the transition
Algorithm 1: ResetTransitionClocks

\[ \text{Algorithm 1: ResetTransitionClocks} \]

**Input:** The choice-point net \( N_{CP} = (P, T, I, C, R, SL, ML, S, MD, CU, M, m_0) \)

A particular node \( \text{thisNode} = (m, \chi, \rho, \theta) \)

1) \( \text{enabledTransitions} \leftarrow \text{GetEnabledTransitions}(N_{CP}, m) \)

// Update the clocks of timer transitions
2) \( \text{forall the } t \in T_R \text{ do} \)
3) \( \text{if } t \notin \text{enabledTransitions then} \)
4) \( \chi(t) \leftarrow -1 \)
5) \( \text{else if } \chi(t) = -1 \text{ then} \)
6) \( \chi(t) \leftarrow R_N(t, \theta) \)
7) \( \text{end} \)
8) \( \text{end} \)

// Update the clocks of single lapse transitions
9) \( \text{forall the } t \in T_{SL} \text{ do} \)
10) \( \text{if } t \notin \text{enabledTransitions then} \)
11) \( \chi(t) \leftarrow -1 \)
12) \( \text{else if } \chi(t) = -1 \text{ then} \)
13) \( \chi(t) \leftarrow SL(t) \)
14) \( \text{end} \)
15) \( \text{end} \)

// Update the clocks of multiple lapse transitions
16) \( \text{forall the } t \in T_{ML} \text{ do} \)
17) \( \text{if } t \notin \text{enabledTransitions then} \)
18) \( \chi(t) \leftarrow \emptyset \)
19) \( \text{else} \)
20) \( \text{// If the transition is newly enabled or has just fired and is} \)
21) \( \text{// still enabled, create lapses for newly-added enabling sets} \)
22) \( \text{while } |\chi(t)| < ED(t, m) \text{ do} \)
23) \( \chi(t) \leftarrow \chi(t) \cup \{B_{ML}(t)\} \)
24) \( \text{end} \)
25) \( \text{end} \)

that is firing is the first in a conflict (line 4); if so, the algorithm reduces the transition clocks by the elapsed time associated with that transition (lines 5-17).
Algorithm 2: ResetFiringNumbers

**Input:** The choice-point net $\mathcal{N}_{CP} = (P, T, I, R, SL, ML, S, MD, CU, M, m_0)$

A particular node $\text{thisNode} = (m, \chi, \rho, \theta)$

// Determine the set of transitions that will fire next based on clock values
1) $\text{nextTransitionsToFire} \leftarrow \text{FindNextToFire}(\mathcal{N}_{CP}, m, \chi)$

// Go through each one of those transitions
2) forall the $t \in \text{nextTransitionsToFire}$ do

3) if $t \in T_{MD}$ then
   // If $t$ is marking-dependent, assign the number produced by its
   // function using the current marking
   4) $md \leftarrow MD(t)$
   5) $\rho(t) \leftarrow md(m)$
   // If necessary, alter the firing number from impossible to possible
   6) if $\rho(t) < 0$ then
      7) $\rho(t) \leftarrow 0$
   else if $\rho(t) > ED(t, m)$ then
      9) $\rho(t) \leftarrow ED(t, m)$
   end

11) else if $t \in T_{ML}$ then
   // If $t$ is multiple lapse, assign the number of minimal lapses
   12) $\rho(t) \leftarrow MLD(\chi(t))$

13) else
   // If $t$ is single lapse, assign the value one
   14) $\rho(t) \leftarrow 1$

end

is also responsible for maintaining the set of transitions that may yet fire. It is possible
during the course of a conflict that the occurrence of some events may prevent others
from happening. This algorithm goes through each transition that is set to fire and
removes those that may no longer do so (lines 22-32). Once the conflict has been
resolved, this algorithm calls Algorithms 1 and 2 to refresh the node that will begin
a new firing sequence (lines 35-38).

Nodes are created by exploring all possible firing sequences in $\mathcal{N}_{CP}$ and generat-
ing identical paths in $\mathcal{RG}_{CP}$. Algorithm 4 is the primary algorithm responsible for
unravelling the graph in this manner by recursively examining each new node, and
Algorithm 3: CreateNextNode

**Input:** The choice-point net $N_{CP} = (P, T, I, C, R, SL, ML, S, MD, CU, M, m_0)$
- The current node $currentNode = (m, \chi, \rho, \theta)$
- The set of transition(s) that are currently attempting to fire $nextTransToFire$
- The transition that is currently firing $t$
- The choice that is currently selected $c$
- The current edge time $edgeTime$

**Output:** The new node $nextNode$

// Create the next transition clock and firing values
1) $nextTransitionClocks \leftarrow$ copy of $\chi$
2) $nextFiringNumbers \leftarrow$ copy of $\rho$

// Reduce the numbers for the transition $t$ that is currently firing
3) $nextFiringNumbers(t) \leftarrow nextFiringNumbers(t) - 1$

// Decrement clocks for lapse transitions that are enabled and counting down
4) if $edgeTime > 0$ then
5)   forall the $t' \in T_{SL}$ do
6)     if $nextTransitionClocks(t') \neq -1$ then
7)       $nextTransitionClocks(t') \leftarrow nextTransitionClocks(t') - edgeTime$
8)     end
9)   end
10)  forall the $t' \in T_{ML}$ do
11)    if $nextTransitionClocks(t') \neq \emptyset$ then
12)       forall the $l \in nextTransitionClocks(t')$ do
13)         $l \leftarrow l - edgeTime$
14)       end
15)    end
16)  end
17)  end

// If $t$ is multiple lapse, remove a zero lapse from the clock set
18) if $t \in T_{ML}$ then
19)   $nextTransitionClocks(t) \leftarrow nextTransitionClocks(t) - \{0\}$
20) end

// Continued on next page ...

all possible child nodes, that are discovered. This algorithm begins by determining whether the given node is part of a conflict or whether a new set of transitions that can fire must be found. To ensure that conflicts are fully explored, all possible sequences of transitions firing and choices made are examined. Each step results in the
// Continuing from previous page ...

// Create the next marking
21) nextMarking ← m' where (m, χ, ρ, θ)(t, c, q, k)(m', χ', ρ', θ')
// Go through the transitions that are attempting to fire
22) for all the t' ∈ nextTransToFire do
    // Check for previously-ready transitions that have been
    // pre-empted by t and c or for those that are done firing
23) if ED(t', nextMarking) = 0 or nextFiringNumbers(t') = 0 then
        // Disable the transition’s clock and firing numbers
        // and remove it from the list
24) if t' ∉ TML then
25)     | nextTransitionClocks(t') ← −1
26) end
27) nextFiringNumbers(t') ← 0
28) nextTransToFire ← nextTransToFire − {t'}
29) else if ED(t', nextMarking) < nextFiringNumbers(t') then
30)     | nextFiringNumbers(t') ← ED(t', nextMarking)
31) end
32) end
33) nextGlobalClock ← (θ + edgeTime) mod CU
34) nextNode ← (nextMarking, nextTransitionClocks, nextFiringNumbers, nextGlobalClock)
35) if nextTransToFire = ∅ then
    // If no transitions are currently set to fire, reset the clocks
    // and the firing numbers for this node
36)     ResetTransitionClocks(NCP, nextNode)
37)     ResetFiringNumbers(NCP, nextNode)
38) end
39) return nextNode

creation of a node and edge. Nodes that have already been located are not recursively
explored but the associated edge is added to the graph. Note that the probability for
that edge is divided by the number of transitions currently in conflict. Nodes that
are not duplicates are the subject of a recursive call.

Algorithm 5 is the method that begins the process by creating an empty ARG
with an initial node and calling Algorithm 4. The initial node is given the initial
marking and global clock value of the net, then submitted to Algorithms 1 and 2
to initialize the transition clock and firing number values. Once the single-node
Algorithm 4: ConstructARGRecursive

**Input:** The choice-point net \( N_{CP} = (P, T, I, C, R, SL, ML, S, MD, CU, M, m_0) \)
- The set of transition(s) that are next to fire \( \text{nextTransitionsToFire} \)
- The current enhanced augmented reachability graph \( R_{G_{CP}} = (\Omega, X, \delta, x_0) \)
- The current node \( \text{currentNode} = (m, \chi, \rho, \theta) \)

1) **if** \( \text{nextTransitionsToFire} = \emptyset \) **then**

   // If no transitions are currently set to fire, find the next set
   // based on both timing and firing values

2) \( \text{nextTransitionsToFire} \leftarrow \text{FindNextToFire}(N_{CP}, m, \chi, \rho) \)

3) \( \text{edgeTime} \leftarrow \text{GetTimeElapsed}(\text{nextTransitionsToFire}, \chi, \theta) \)

4) **else**

   // Otherwise, a conflict is currently underway

5) \( \text{edgeTime} \leftarrow 0 \)

6) **end**

// Fire each transition that is ready
7) **for all** \( t \in \text{nextTransitionsToFire} \) **do**

   // Explore all the choices for this transition

8) **for all** \( c = (l, b, O) \in C_T(t) \) **do**

   // Create the set of transitions to fire for the
   // recursive call

9) \( \text{recTransitionsToFire} \leftarrow \text{copy of nextTransitionsToFire} \)

   // Create the child node and add it to the tree

10) \( \text{nextNode} \leftarrow \text{CreateNextNode}(N_{CP}, \text{currentNode}, \text{recTransitionsToFire}, t, c, \text{edgeTime}) \)

11) \( \text{nextProbability} \leftarrow b \div b_{\text{nextTransitionsToFire}} \)

12) \( \text{nextEdge} \leftarrow (t, c, \text{nextProbability}, \text{edgeTime}) \)

13) \( \delta(\text{currentNode}, \text{nextEdge}) \leftarrow \text{nextNode} \)

14) **if** \( \text{nextNode} \notin X \) **then**

   15) \( X \leftarrow X \cup \{\text{nextNode}\} \)

   16) **ConstructARGRecursive**\( (N_{CP}, \text{recTransitionsToFire}, R_{G_{CP}}, \text{nextNode}) \)

17) **end**

18) **end**

19) **end**

augmented reachability graph is created, Algorithm 4 does the rest.
Algorithm 5: ConstructARG

Input: A choice-point net $N_{CP} = (P, T, I, C, R, SL, ML, S, MD, CU, M, m_0)$
An initial global clock value $0 \leq GCV < CU$
Output: An augmented reachability graph $RG_{CP}$

1) $\chi \leftarrow$ a new transition clock function where for all $t \in (T_R \cup T_{SL})$, $\chi(t) = -1$ and for all $u \in T_{ML}$, $\chi(u) = \emptyset$
2) $\rho \leftarrow$ a new firing number function where for all $t \in T$, $\rho(t) = 0$
3) $\delta \leftarrow$ a new edge function with no entries
4) $\text{initialNode} \leftarrow (m_0, \chi, \rho, GCV)$
5) ResetTransitionClocks($N_{CP}$, $\text{initialNode}$)
6) ResetFiringNumbers($N_{CP}$, $\text{initialNode}$)
7) $RG_{CP} \leftarrow (\Omega, \{\text{initialNode}\}, \delta, \text{initialNode})$
8) ConstructARGRecursive($N_{CP}$, $\emptyset$, $RG_{CP}$, $\text{initialNode}$)
9) return $RG_{CP}$

4.5 An Example of ARG Construction

To illustrate how the algorithms defined in the previous section build an augmented reachability graph, the following “walkthrough” is provided. It is an in-depth examination of part of the construction of the graph shown in Figure 4.2.

Consider the node with marking $m = [1\ 1\ 0]$, $\chi(s\text{SeeIIIResident}) = 8$, $\chi(r\text{DonelIsolation}) = \{9\}$, $\rho(s\text{SeeIIIResident}) = \rho(r\text{DonelIsolation}) = 1$ and $\theta = 23$.

Algorithm 4 is called with this node as $\text{currentNode}$ and $\text{nextTransitionsToFire} = \emptyset$.

Lines 1–6 set $\text{nextTransitionsToFire} = \{s\text{SeeIIIResident}, r\text{DonelIsolation}\}$ and $\text{edgeTime} = 9$. Taking $t = s\text{SeeIIIResident}$ on line 7, $c = (\text{Isolate}, 0.8, O)$ on line 8 and $\text{recNextTransitionsToFire} = \{s\text{SeeIIIResident}, r\text{DonelIsolation}\}$ on line 9, Algorithm 3 is called on line 10.

During the call to Algorithm 3, lines 1–2 copy $\chi$ and $\rho$ into $\text{nextTransitionClocks}$ and $\text{nextFiringNumbers}$, respectively. Line 3 sets $\text{nextFiringNumbers}(s\text{SeeIIIResident}) = 0$. Since $\text{edgeTime} = 9$, the $\text{if}$ statement on lines 4–17 is entered. There are
no single lapse transitions in this net, so the `forall` loop on lines 5–9 is ignored. However, transition `rDoneIsolation` is examined within the `forall` on lines 10–16; `nextTransitionClocks(rDoneIsolation) = {0}` once the `forall` statement on lines 12–14 is complete. Since \( t = s\text{SeellIResident} \) is not a multiple lapse transition, the `if` statement on lines 18–20 is ignored.

The new marking \([0 \ 2 \ 0]\) is created on line 21 and assigned to variable `nextMarking`. The `forall` loop on lines 22–32 is then entered. For the first iteration, \( t' = s\text{SeellIResident} \) and in this case `nextFiringNumbers(s\text{SeellIResident}) = 0`, satisfying the `if` condition on line 23. Since `s\text{SeellIResident}` is not a multiple lapse transition, the `if` statement on lines 24–26 is entered and `nextTransitionClocks(s\text{SeellIResident}) = -1` on completion of line 25. Redundantly, `nextFiringNumbers(s\text{SeellIResident})` is assigned a value of zero on line 27 and `nextTransitionsToFire = \{r\text{DoneIsolation}\}` once line 28 is complete. For the second iteration, \( t' = r\text{DoneIsolation} \) and in this case \( ED(t', \text{nextMarking}) = 2 \) while `nextFiringNumbers(s\text{SeellIResident}) = 1`. Neither of the conditions on lines 23 and 29 are satisfied, ending the final iteration of this loop.

The new global clock value \((23 + 9) \mod 24 = 8\) is assigned to `nextGlobalClock` on line 33; `nextNode` is created from all of the `next` variables on line 34. Since `nextTransitionsToFire = \{r\text{DoneIsolation}\} \neq \emptyset`, the `if` statement on lines 35–38 is never entered. This call to Algorithm 3 therefore ends and control is returned to Algorithm 4.

With `nextNode` complete in Algorithm 4, line 11 sets the probability `nextProbability` of this choice to \(0.8 \div 2 = 0.4\) because a conflict is underway between two transitions. Line 12 creates the edge `nextEdge` from \( t, c, \text{nextProbability} \) and `edgeTime` while line 13 makes the entry in the transition function for `currentNode`, `nextEdge` and `nextNode`. 
At this point, $\text{nextNode} \notin X$ and so the if statement on lines 14–17 is entered. The node is added to the set on line 15 and a recursive call is made on line 16.

In this recursive call to Algorithm 4, things proceed differently because $\text{nextTransitionsToFire} = \{\text{rDonelIsolation}\}$. The elements of $\text{currentNode}$ are now $m = [0 \ 2 \ 0]$, $\chi(s\text{SeellIIResident}) = -1$, $\chi(\text{rDonelIsolation}) = \{0\}$, $\rho(s\text{SeellIIResident}) = 0$, $\rho(\text{rDonelIsolation}) = 1$ and $\theta = 8$. Lines 1–6 set $\text{edgeTime} = 0$. Taking $t = \text{rDonelIsolation}$ on line 7, $c = (\text{rDI}, 1.0, O)$ on line 8 and $\text{recNextTransitionsToFire} = \{\text{rDonelIsolation}\}$ on line 9, Algorithm 3 is called on line 10.

Within Algorithm 3, lines 1–2 again copy $\chi$ and $\rho$ into $\text{nextTransitionClocks}$ and $\text{nextFiringNumbers}$, respectively, while line 3 sets $\text{nextFiringNumbers}(\text{rDonelIsolation}) = 0$. In this case $\text{edgeTime} = 0$ and the if statement on lines 4–17 is avoided. However, since $t = \text{rDonelIsolation}$ is a multiple lapse transition, the if statement on lines 18–20 is entered and $\text{nextTransitionClocks}(\text{rDonelIsolation}) = \emptyset$ once line 19 is complete.

The new marking is $[0 \ 1 \ 1]$ and is created on line 21; the forall loop on lines 22–32 is then entered. The only iteration sees $t' = \text{rDonelIsolation}$ and in this case $\text{nextFiringNumbers}(\text{rDonelIsolation}) = 0$, satisfying the if condition on line 23. Since $\text{rDonelIsolation}$ is a multiple lapse transition, the if statement on lines 24–26 is ignored. Again, $\text{nextFiringNumbers}(\text{rDonelIsolation})$ is redundantly assigned a value of zero on line 27 and $\text{nextTransitionsToFire} = \emptyset$ once line 28 is complete.

The next global clock value is calculated as $(8 + 0) \mod 24 = 8$ on line 33 and $\text{nextNode}$ is created on line 34. In this case $\text{nextTransitionsToFire} = \emptyset$ and the if block on lines 35–38 is entered. Algorithms 1 and 2 are called on lines 36 and 37, respectively, setting $\text{nextTransitionClocks}(\text{rDonelIsolation}) = \{24\}$ and
nextFiringNumbers(rDoneIsolation) = 1. Afterwards, this call to Algorithm 3 ends and control is returned to Algorithm 4.

Now that nextNode has been created, line 11 in Algorithm 4 sets nextProbability to 1.0 ÷ 1 = 1.0 now that the conflict has been resolved. Line 12 creates the edge nextEdge and line 13 creates the transition function entry for currentNode, nextEdge and nextNode. At this stage of the exploration, nextNode \( \notin X \) and the if block on lines 14–17 is entered; nextNode is added to the graph on line 15 and a recursive call is made on line 16.
Chapter 5

Behavioural Equivalence

It is necessary to demonstrate that the algorithms used to construct an augmented reachability graph from a given choice-point net do, indeed, preserve the behaviour captured by the net that gave rise to the graph. This thesis does so by showing that for any sequence of events and selection of choices in $\mathcal{N}_{CP}$ there exists a corresponding path in the generated $\mathcal{RG}_{CP}$ whose marking and timing values are identical. Similarly, the elements of any path in the $\mathcal{RG}_{CP}$ will match a firing in the $\mathcal{N}_{CP}$. These two statements are formally defined and proven in Theorem 1, which will appear later in this chapter. First, the properties of Algorithms 1–4 must be established.

5.1 Algorithm 1

The algorithm ResetTransitionClocks is designed to alter the transition clock values of $\chi$ in thisNode following the firing of at least one (and perhaps many) transition(s) in $\mathcal{N}_{CP}$. The node thisNode contains the updated marking $m$ and global clock value $\theta$. The firing numbers $\rho$ are not examined here; $\chi$ must be updated first as the
firing numbers rely on this value for their own reset. The purpose of Algorithm 1 is to bring $\chi$ up to speed with respect to transitions that have just fired and transitions that are now disabled.

This algorithm handles three of the four transition clock cases for each transition $t \in T$ in the choice-point net $N_{CP}$; the other is handled by Algorithm 3. The first case is that $t$ may be disabled in the new marking $m$—these transitions must have their clocks disabled, represented by an empty set for multiple lapse transitions or the value $-1$ for all other types.

**Lemma 1-1.** Upon the completion of Algorithm 1, $\chi(t) = -1$ for all transitions $t \in (T_R \cup T_{SL})$ where $ED(t,m) = 0$ (i.e., that are disabled under marking $m$) in choice-point net $N_{CP}$. Similarly, $\chi(u) = \emptyset$ for all transitions $u \in T_{ML}$ that are disabled under $m$.

**Proof.** The set of enabled transitions in $N_{CP}$ under marking $m$ is assigned to enabledTransitions on line 1. All timer, single lapse and multiple lapse transitions in $N_{CP}$ are examined in the forall loops on lines 1–8, 9–15 and 16–24, respectively. All disabled timer and single lapse transitions (i.e., those not in enabledTransitions) are given timing values of $-1$ on lines 3–4 and 10–11, respectively. All disabled multiple lapse transitions are given timing values of $\emptyset$ on lines 17–18.

The second transition clock case involves timer and single lapse transitions that must have their clock values reset. These transitions are those that were disabled under the previous marking but are newly enabled under $m$, or those that most recently fired but are enabled once again in $m$. For a lapse transition $t_l \in T_{SL}$, this means that its transition clock must be set to its full lapse value (e.g., $SL(t_l) = 4$). For
a timer transition $u_r \in T_R$, its transition clock must be set to whichever of its possible values will occur in the fewest number of clock ticks (e.g., if $R(u_r) = \{4, 11, 15\}$ and $\theta = 12$, then $\chi(u_r) = 15$).

**Lemma 1-2.** For all transitions $t \in T_R \cup T_{SL}$ where $\chi(t) = -1$ and $ED(t, m) > 0$ (i.e., $t$ is enabled under marking $m$), if $t \in T_{SL}$ then $\chi(t) = SL(t)$ and if $t \in T_R$ then $\chi(t) = R_N(t, \theta)$ upon the completion of Algorithm 1.

**Proof.** All timer and single lapse transitions in $N_{CP}$ are examined in the forall loops on lines 1–8 and 9–15, respectively. Any such transition $t$ that is enabled under the given marking (i.e., in enabledTransitions) and has a transition clock value of $-1$ (lines 5 and 12) has either its next timer value (line 6) or a new lapse (line 13) assigned to its transition clock $\chi(t)$.

The third transition clock case is concerned with multiple lapse transitions whose input places are populated with new enabling sets, deposited as a result of the previous firing sequence. New lapses must be created and initialized for each new set of tokens. For example, consider a multiple lapse transition $t$ where $\chi(t) = \{9, 15\}$ and $B_{ML}(t) = 24$. If the enabling degree of $t$ under $m$ is now four, two new individual lapses of 24 must be added to $\chi(t)$ to begin the countdown for the recently added tokens, resulting in $\chi(t) = \{9, 15, 24, 24\}$.

**Lemma 1-3.** Let previousSet = $\chi(t)$ at the beginning of Algorithm 1. For all transitions $t \in T_{ML}$ where $ED(t, m) > |\text{previousSet}|$, upon the completion of Algorithm 1, $ED(t, m) = |\chi(t)|$ and for all $x \in (\chi(t) - \text{previousSet})$, $x = B_{ML}(t)$.

**Proof.** All multiple lapse transitions in $N_{CP}$ are examined in the forall loop on lines 16–24. If $t$ is not disabled (tested on line 17), the else statement on lines 19–23 is
Chapter 5. Behavioural Equivalence

entered. The while loop on lines 20–22 repeatedly adds values of $B_{ML}(t)$ to $\chi(t)$ until its size matches the enabling degree.

The previous lemmas have established the functionality of Algorithm 1—all that remains is to connect and formalize these properties into a single proposition.

**Proposition 1.** Consider a choice-point net $N_{CP}$ in state $(m_{\text{pre}}^{N_{CP}}, x_{\text{pre}}^{N_{CP}}, \rho_{\text{pre}}^{N_{CP}}, \theta_{\text{pre}}^{N_{CP}})$ with transitions $\{t_1, \ldots, t_n\} \subseteq T$ ready to fire. Once all of these transitions have either fired or been pre-empted, $N_{CP}$ is in state $(m, x_{\text{post}}^{N_{CP}}, \rho_{\text{post}}^{N_{CP}}, \theta)$. Assume the following about $\text{thisNode} = (m, x, \rho, \theta)$:

- for all $t \in \{t_1, \ldots, t_n\} \cap (T_R \cup T_{SL})$, $\chi(t) = -1$
- for all $t_{sl} \in (T_{SL} - \{t_1, \ldots, t_n\})$ where $\chi(t_{sl}) \neq -1$, $\chi(t_{sl}) = x_{\text{post}}^{N_{CP}}(t_{sl})$
- for all $t_{ml} \in T_{ML}$, $\chi(t_{ml}) \subseteq x_{\text{post}}^{N_{CP}}(t_{ml})$

Upon the completion of this algorithm, $\text{thisNode}$ contains reset transition clock values that are consistent with those maintained by $N_{CP}$. Formally, for all $t \in T$, $\chi(t) = x_{\text{post}}^{N_{CP}}(t)$.

**Proof.** Follows from Lemmas 1-1, 1-2 and 1-3.

In addition, Algorithm 1 can be called to initialize the transition clock values $\chi$ for an initial marking $m_0$ and initial global clock value $GCV$. The following proposition proves that the results are accurate.

**Proposition 2.** Consider a choice-point net $N_{CP}$ in initial state $(m_0^{N_{CP}}, x_{\text{init}}^{N_{CP}}, \rho_{\text{init}}^{N_{CP}}, GCV)$. Let $\text{thisNode} = (m_0, x, \rho, GCV)$ where for all $t \in (T_R \cup T_{SL})$, $\chi(t) = -1$ and for all $u \in T_{ML}$, $\chi(u) = \emptyset$. Upon the completion of Algorithm 1, $\text{thisNode}$ contains
transition clock values that are consistent with those initialized by \( N_{CP} \). Formally, for all \( t \in T \), \( \chi(t) = \chi_{\text{init}}^{N_{CP}}(t) \).

**Proof.** Follows from Lemmas 1-1, 1-2 and 1-3. \( \square \)

## 5.2 Algorithm 2

The algorithm \( \text{ResetFiringNumbers} \) is similar in purpose to \( \text{ResetTransitionClocks} \) —it updates the firing number values of \( \rho \) in \( \text{thisNode} \) following the firing of some transition(s) in \( N_{CP} \). To determine the correct firing values, the node \( \text{thisNode} \) must contain updated marking \( m \) and transition clock values \( \chi \), which means that Algorithm 1 must be called beforehand. Algorithm 2 is responsible for calculating the number of times the transitions whose clock values permit them to fire next may each do so.

**Proposition 3.** Consider a choice-point net \( N_{CP} \) in state \((m_{\text{pre}}^{N_{CP}}, \chi_{\text{pre}}^{N_{CP}}, \rho_{\text{pre}}^{N_{CP}}, \theta_{\text{pre}}^{N_{CP}})\) with transitions \( \{t_1, \ldots, t_n\} \subseteq T \) ready to fire. Once all of these transitions have either fired or been pre-empted, \( N_{CP} \) is in state \((m, \chi, \rho_{\text{post}}^{N_{CP}}, \theta)\). Assume the following about \( \text{thisNode} = (m, \chi, \rho, \theta) \):

- for all \( t \in \{t_1, \ldots, t_n\} \), \( \rho(t) = 0 \)

Upon the completion of Algorithm 2, \( \text{thisNode} \) contains reset firing number values that are consistent with those maintained by \( N_{CP} \). Formally, for all \( t \in T \), \( \rho(t) = \rho_{\text{post}}^{N_{CP}}(t) \).

**Proof.** The set of transitions in \( N_{CP} \) that are next to fire given current marking \( m \) and transition clock values \( \chi \) is assigned to \( \text{nextTransitionsToFire} \) on line 1 via helper method \( \text{FindNextToFire} \). All transitions in this set are examined in the forall loop
Chapter 5. Behavioural Equivalence

on lines 2–11. If \( t \) is marking-dependent (line 3), the marking-dependent function returned by \( MD(t) \) is applied to marking \( m \) with the result assigned to \( \rho(t) \) (lines 4–5). If necessary, this value is adjusted to fit within the range of possible firing values \([0, ED(t, m)]\) (lines 6–10). If \( t \) is a multiple lapse transition (line 11), the number of identical minimum lapse values in \( \chi(t) \) is assigned to \( \rho(t) \) (line 12) by the helper function \( MLD \). Finally, if \( t \) is single lapse (line 13) then \( \chi(t) \) is assigned a value of 1 (line 14) because, by definition, a single lapse transition may only fire once. \( \square \)

As with Algorithm 1, Algorithm 2 may be called to initialize the firing numbers \( \rho \) for an initial marking \( m_0 \), global clock value \( GCV \) and initialized transition clock values \( \chi \).

**Proposition 4.** Consider a choice-point net \( N_{CP} \) in initial state \((m_0, \chi_{init}^{N_{CP}}, \rho_{init}^{N_{CP}}, GCV)\). Let thisNode = \((m_0, \chi, \rho, GCV)\) where for all \( t \in T \), \( \rho(t) = 0 \). Upon the completion of Algorithm 2, thisNode contains firing number values that are consistent with those initialized by \( N_{CP} \). Formally, for all \( t \in T \), \( \rho(t) = \rho_{init}^{N_{CP}}(t) \).

**Proof.** Identical to that for Proposition 3. \( \square \)

### 5.3 Algorithm 3

The purpose of algorithm `CreateNextNode` is to determine the next node in \( \mathcal{RG}_{CP} \) and the next set of transitions to fire after the occurrence of a transition \( t \in T \) and the selection of a choice \( c \in C_T(t) \). The current state of \( N_{CP} \) is assumed to be captured by `currentNode` with marking \( m \), transition clocks \( \chi \), firing numbers \( \rho \) and global clock \( \theta \). The parameter `edgeTime` is responsible for storing the time elapsed between the occurrence of the last transition to fire and \( t \) itself. However, \( t \) may not be the first
to fire in the conflict; there may actually be no delay between it firing and any other transition in \texttt{nextTransitionsToFire} firing. The value of \texttt{edgeTime} is assumed to be nonzero if \( t \) is the first selected from \texttt{nextTransitionsToFire} and zero otherwise. This algorithm must return a new and fully updated node; the marking, transition clocks, firing numbers and global clock values must all reflect the impact of \( t \) and \( c \) on the system.

The first task is to copy the current transition clock values and firing numbers as a starting point for the new node; they will be adjusted by the remainder of the algorithm. In addition, the firing number for \( t \) is decremented here to record the occurrence of this event.

**Lemma 3-1.** Let \texttt{currentNode} represent the current state \((m, \chi, \rho, \theta)\) of a choice-point net \( N_{CP} \) with the transitions in \texttt{nextTransitionsToFire} left to fire. Upon the conclusion of line 3 of Algorithm 3, for all \( u \in T \), \( \text{nextTransClocks}(u) = \chi(u) \) while for all \( v \in T - \{t\} \), \( \text{nextFiringNumbers}(v) = \rho(v) \) and \( \text{nextFiringNumbers}(t) = \rho(t) - 1 \).

**Proof.** The transition clock mapping \texttt{nextTransClocks} is copied from \( \chi \) on line 1; the same can be said for the firing number mapping \texttt{nextFiringNumbers} and \( \rho \) on line 2. The firing number associated with \( t \) in \texttt{nextFiringNumbers} is decremented on line 3. \( \square \)

It was stated previously that Algorithm 1 handles three of the cases with respect to updating transition clocks; the other is managed here. The fourth case involves the net \( N_{CP} \) firing the first transition in a conflict sequence (or perhaps alone). In this case, \texttt{edgeTime} has a nonzero value. To reflect the change in time, all single lapse transitions \( t_{sl} \in T_{SL} \) and multiple lapse transitions \( t_{ml} \in T_{ML} \) that are enabled under
the current marking must have their transition clock values updated. Specifically, the single countdown for \( t_{sl} \) and the multiple countdowns for \( t_{ml} \) must all be decremented by \( \text{edgeTime} \) clock units. It is assumed that \( \text{edgeTime} \) is less than or equal to any current lapse value in \( \chi \) so that the subtraction does not result in any negative values, only zero values for those lapse transitions that are currently firing, \( i.e. \), those in \( \text{nextTransitionsToFire} \).

Note that transition clock values for timer transitions \( u_r \in T_R \) that are enabled are not altered. Timer transitions must retain their current values because they still have to wait to match a future time on the global clock; they are not counting down like lapses.

**Lemma 3-2.** Let \( \text{edgeTime} > 0 \) and assume the following:

- for all \( t_{sl} \in (T_{SL} - \text{nextTransitionsToFire}) \) where \( \chi(t_{sl}) \neq -1 \), \( \text{edgeTime} < \chi(t_{sl}) \)
- for all \( t_{sl} \in (T_{SL} \cap \text{nextTransitionsToFire}) \), \( \chi(t_{sl}) = \text{edgeTime} \)
- for all \( u_{ml} \in (T_{ML} - \text{nextTransitionsToFire}) \) where \( \chi(u_{ml}) \neq \emptyset \), for all \( x \in \chi(u_{ml}) \), \( \text{edgeTime} < x \)
- for all \( u_{ml} \in (T_{ML} \cap \text{nextTransitionsToFire}) \), for all \( x \in \chi(u_{ml}) \), \( \text{edgeTime} \leq x \) and there exists \( y \in \chi(u_{ml}) \) where \( \text{edgeTime} = y \)

Upon the completion of line 17 of Algorithm 3,

- for all \( t_{sl} \in (T_{SL} - \text{nextTransitionsToFire}) \) where \( \text{nextTransitionClocks}(t_{sl}) \neq -1 \), \( \text{nextTransitionClocks}(t_{sl}) = \chi(t_{sl}) - \text{edgeTime} \) and hence \( \text{nextTransitionClocks}(t_{sl}) > 0 \)
- for all \( t_{sl} \in (T_{SL} \cap \text{nextTransitionsToFire}) \), \( \text{nextTransitionClocks}(t_{sl}) = 0 \)
• for all \( u_{ml} \in (T_{ML} - \text{nextTransitionsToFire}) \) where \( \text{nextTransitionClocks}(u_{ml}) \neq \emptyset \), if \( \chi(u_{ml}) = \{x_1, \ldots, x_n\} \) then \( \text{nextTransitionClocks}(u_{ml}) = \{(x_1 - \text{edgeTime}), \ldots, (x_n - \text{edgeTime})\} \) and \( (x_i - \text{edgeTime}) > 0 \) for all \( 1 \leq i \leq n \)

• for all \( u_{ml} \in (T_{ML} \cap \text{nextTransitionsToFire}) \), if \( \chi(u_{ml}) = \{x_1, \ldots, x_n\} \) then \( \text{nextTransitionClocks}(u_{ml}) = \{(x_1 - \text{edgeTime}), \ldots, (x_n - \text{edgeTime})\} \) where \( (x_i - \text{edgeTime}) \geq 0 \) for all \( 1 \leq i \leq n \) and \( x_j = 0 \) for some value(s) \( 1 \leq j \leq n \)

Proof. The if statement that requires \( \text{edgeTime} \) to be greater than zero takes place over lines 4–17. All single lapse transitions in the net are examined in the forall loop on lines 5–9; all enabled transitions (i.e., \( \text{nextTransitionClocks}(t') \neq -1 \)) are examined on lines 6–8. These transitions have their values decremented by \( \text{edgeTime} \) on line 7. From the assumptions made about \( \text{edgeTime} \), any lapse transitions that are not currently firing have positive transition clock values while lapse transitions that are currently firing are given zero values.

Similarly, all multiple lapse transitions are considered in the forall loop on lines 10–16; all enabled transitions (i.e., \( \text{nextTransitionClocks}(t') \neq \emptyset \)) are considered on lines 11–15. Each lapse in the set is examined on lines 12–14 and has its value decremented by \( \text{edgeTime} \) on line 13. Again, from the assumptions about \( \text{edgeTime} \) all lapses for transitions that are not currently firing have positive transition clock values while those that are currently firing will have at least one zero value (or possibly many).

Once the transition clocks have been updated in the manner described by Lemma 3-2, a multiple lapse transition in \( \text{nextTransitionsToFire} \) will contain a lapse with value 0 for each time it must fire. If the transition \( t \) that is currently firing is multiple lapse, a zero lapse must be removed.
Lemma 3-3. Let $currentNode$ represent the current state $(m, \chi, \rho, \theta)$ of a choice-point net $N_{CP}$ with the transitions in $nextTransitionsToFire$ left to fire and let $previousValue = nextTransitionClocks$ prior to line 19. If $t \in T_{ML}$, upon the conclusion of line 20 of Algorithm 3, $nextTransitionClocks(t) = previousValue - \{0\}$.

Proof. The if statement that tests whether $t$ is a multiple lapse transition occurs on line 18; if it is, a single zero lapse is removed on line 19.

The next task when creating a new node is to determine the new marking and global clock following the firing of a transition and the selection of a choice—the following lemma shows that Algorithm 3 performs this function.

Lemma 3-4. Let $currentNode$ represent the current state $(m, \chi, \rho, \theta)$ of a choice-point net $N_{CP}$ with the transitions in $nextTransitionsToFire$ currently attempting to fire. Let $edgeTime > 0$ if $t \in T$ is the first to fire from $nextTransitionsToFire$; let $edgeTime = 0$ otherwise. Upon the conclusion of Algorithm 3, $nextNode$ contains the next marking $nextMarking$ where $(m, \chi, \rho, \theta))(t, c, p, k))(nextMarking, \chi', \rho', \theta')$ in $N_{CP}$ and the next global clock value $nextGlobalClock = (\theta + edgeTime) \mod CU$.

Proof. The next marking is determined from the net using the given transition and choice on line 21. The subsequent time on the global clock is calculated with $edgeTime$, taking rollover into account using modulo the number of clock units $CU$ on line 33. The new node $nextNode$ is then created with these elements on line 34.

One of the primary functions of Algorithm 3 is to verify that all the transitions in $nextTransitionsToFire$ may still fire according to their firing numbers $nextFiringNumbers$. For example, it is possible that this call to the algorithm represents the last time
t may fire (i.e., $\text{nextFiringNumbers}(t) = 0$). However, in the event of a conflict where multiple transitions are attempting to fire, it is also possible that this occurrence of $t$ may affect the ability of the other transitions in $\text{nextTransitionsToFire}$ to actually fire.

Consider two competing transitions $u, v \in \text{nextTransitionsToFire}$ that are in pre-emptive conflict. If $u$ occurs first, it will “steal” some of the tokens necessary for $v$ to fire; the same could be said for $v$ and $u$, respectively. It is therefore necessary to scan $\text{nextTransitionsToFire}$, remove any transitions that may have been entirely pre-empted by the occurrence of $t$ and adjust the firing numbers for transitions that have had some (but not all) of their enabling sets removed by $t$. It is important to note that pre-emption is impossible for multiple lapse transitions as they are assumed to not be in direct conflict with any other transitions. Moreover, the transition clock values for multiple lapse transitions are not disabled here in case there are additional nonzero lapses that must still count down.

**Lemma 3-5.** Let $\text{currentNode}$ represent the current state $(m, \chi, \rho, \theta)$ of a choice-point net $N_{CP}$ with the transitions in $\text{nextTransitionsToFire}$ currently attempting to fire. From Lemma 3-4, $\text{nextMarking}$ contains the marking that results from the firing of $t$ and the selection of $c \in C_T(t)$. Upon the conclusion of line 32 of Algorithm 3,

- For all transitions $u \in \text{nextTransitionsToFire}$ where $ED(u, \text{nextMarking}) = 0$ or $\text{nextFiringNumbers}(u) = 0$, $u \notin \text{nextTransitionsToFire}$, $\text{nextFiringNumbers}(u) = 0$ and if $u \notin T_{ML}$, $\text{nextTransitionClocks}(u) = -1$

- For all transitions $v \in \text{nextTransitionsToFire}$ where $\text{nextFiringNumbers}(v) > ED(v, \text{nextMarking})$, $\text{nextFiringNumbers}(v) = ED(v, \text{nextMarking})$

**Proof.** The forall statement that examines all transitions $t' \in \text{nextTransitionsToFire}$ takes place on lines 22–32; all newly disabled transitions (i.e., $ED(t', \text{nextMarking}) =$
0) and transitions that have finished firing (possibly $t$) are examined on lines 23–28. If $t'$ is not multiple lapse (lines 24–26), its transition clock is disabled (i.e., set to -1) on line 25. The firing number for $t'$ is set to zero (possibly redundantly) on line 27 and the transition is removed from nextTransitionsToFire on line 28. Conversely, a transition $t'$ whose enabling degrees is now less than the number of times it must fire is examined on lines 29-31; the transition’s firing number is assigned to the new enabling degree on line 30.

Once all of the transitions in nextTransitionsToFire have either fired or been preempted, the last node generated from a conflict must have its transition clocks and firing numbers reset to determine which transition(s) will fire next and how often. This requires calls to Algorithms 1 and 2 once the last node has been generated. The end of a conflict is represented by nextTransitionsToFire lacking any elements, indicating that all the transitions have finished firing. If this is the case, Algorithm 1 is called on line 36 followed by Algorithm 2 on line 37 due to the if condition on lines 35–38.

The next lemma is concerned with how Algorithm 3 manages the clock values and firing numbers of each transition that is firing. This algorithm is designed to be called repeatedly on a set of conflict transitions (i.e., nextTransitionsToFire) that diminishes as its members individually fire and possibly preempt other transitions in the set. Every conflict node that is generated and all remaining enabled transitions that may still fire should be given as parameters in subsequent calls to Algorithm 3 to accurately track the conflict. In this manner, every firing transition will have its transition clock adjusted and its firing numbers gradually reduced until they are zero.

For example, consider a node $x \in X$ where $\{t_1, t_2, t_3\} \subset T$ may each fire once but
$t_2$ and $t_3$ are in pre-emptive conflict over a single token. A call to Algorithm 3 with $\text{currentNode} = x$, $\text{nextTransitionsToFire} = \{t_1, t_2, t_3\}$ and $t = t_1$ will return a node $y \in X$. A subsequent call to Algorithm 3 should then be made with $\text{currentNode} = y$, $\text{nextTransitionsToFire} = \{t_2, t_3\}$ and $t = t_2$, the occurrence of which pre-empt $t_3$ and thus completes this particular firing sequence. Since it is assumed that any transition may go first, many additional calls should be made to Algorithm 3 to ensure all possible sequences of events in this conflict (i.e., $t_1t_3$, $t_2t_1$ and $t_3t_1$) are explored.

In addition, it is vital to consider that Algorithm 3 calls Algorithms 1 and 2 (on lines 36-37) at the end of a conflict (i.e., when $\text{nextTransitionsToFire} = \emptyset$ as tested on line 35) and must therefore satisfy the conditions of Propositions 1 and 3. If Algorithm 3 is called repeatedly in the aforementioned manner, then for all $v \in \text{nextTransitionsToFire}$, $\rho(v) = 0$ (satisfying the single condition for Proposition 3) and for all $u \in (\text{nextTransitionsToFire} \cap (T_R \cup T_{SL})), \chi(u) = -1$ (satisfying the first condition for Proposition 1). The latter two conditions for Proposition 1 are satisfied by Lemmas 3-2 and 3-3 when Algorithm 3 is called repeatedly in the aforementioned manner.

**Lemma 3-6.** Let $\text{currentNode} = (m, \chi, \rho, \theta)$, $\text{edgeTime} > 0$ and $\text{nextTransitionsToFire} = \{t_1, t_2, \ldots, t_n\} \subseteq T$. Let Algorithm 3 be called repeatedly using $\text{currentNode}$ set to the resulting $\text{nextNode}$ of the previous iteration, an identical set of $\text{nextTransitionsToFire}$ resulting from the previous iteration and some $t \in \text{nextTransitionsToFire}$ as parameters. When $\text{nextTransitionsToFire} = \emptyset$ upon the completion of line 32 of Algorithm 3, for all $u \in (\{t_1, t_2, \ldots, t_n\} \cap (T_R \cup T_{SL})), \text{nextTransitionClocks}(u) = -1$ and for all $v \in \{t_1, t_2, \ldots, t_n\}$, $\text{nextFiringNumbers}(v) = 0$.

**Proof.** If Algorithm 3 is called in the aforementioned manner, then each transition
Chapter 5. Behavioural Equivalence 64

$t_i$ in \{\(t_1, t_2, \ldots, t_n\)\} for \(1 \leq i \leq n\) will have its firing number decremented as stated in Lemma 3-1. When the firing number for any \(t_i\) reaches zero (through firing or preemption), it is removed from the set \texttt{nextTransitionsToFire} and, if \(t_i\) is not a multiple lapse transition, its clock value is set to \(-1\) according to Lemma 3-5. Therefore, when \texttt{nextTransitionsToFire} = \(\emptyset\) all transitions in \{\(t_1, t_2, \ldots, t_n\)\} must have had their transition clock values and firing numbers altered in this manner. 

The preceding lemmas have demonstrated how Algorithm 3 is designed to create new nodes; these properties are formally combined in Proposition 5.

**Proposition 5.** Consider a choice-point net \(N_{CP}\) in state \((m_{pre}^{N_{CP}}, \chi_{pre}^{N_{CP}}, p_{pre}^{N_{CP}}, \theta_{pre}^{N_{CP}})\) and transitions \texttt{nextTransitionsToFire} = \{\(t_1, t_2, \ldots, t_n\)\} \subseteq T enabled after the occurrence of \texttt{edgeTime} clock ticks (where \texttt{edgeTime} > 0). Let \texttt{currentNode} = \((m_{pre}^{N_{CP}}, \chi_{pre}^{N_{CP}}, p_{pre}^{N_{CP}}, \theta_{pre}^{N_{CP}})\) and make the following assumptions:

- For all \(t_{sl} \in (T_{SL} - \texttt{nextTransitionsToFire})\) where \(\chi_{pre}^{N_{CP}}(t_{sl}) \neq -1\), \(\texttt{edgeTime} < \chi_{pre}^{N_{CP}}(t_{sl})\)
- For all \(t_{sl} \in (T_{SL} \cap \texttt{nextTransitionsToFire})\), \(\chi_{pre}^{N_{CP}}(t_{sl}) = \texttt{edgeTime}\)
- For all \(u_{ml} \in (T_{ML} - \texttt{nextTransitionsToFire})\) where \(\chi_{pre}^{N_{CP}}(u_{ml}) \neq \emptyset\), for all \(x \in \chi_{pre}^{N_{CP}}(u_{ml})\), \(\texttt{edgeTime} < x\)
- For all \(u_{ml} \in (T_{ML} \cap \texttt{nextTransitionsToFire})\), for all \(x \in \chi_{pre}^{N_{CP}}(u_{ml})\), \(\texttt{edgeTime} \leq x\) and there exists \(y \in \chi_{pre}^{N_{CP}}(u_{ml})\) such that \(\texttt{edgeTime} = y\)
- Algorithm 3 is called repeatedly using \texttt{currentNode} set to the resulting \texttt{nextNode} of the previous iteration, an identical set of \texttt{nextTransitionsToFire} resulting from the previous iteration and some \(t \in \texttt{nextTransitionsToFire}\) as parameters
Let \((m, \chi, \rho, \theta)\) represent a possible subsequent state of \(N_{CP}\) within this conflict. The firing of some \(t \in \text{nextTransitionsToFire}\) and the selection of some choice \(c \in C_T(t)\) results in the net entering state \((m_{post}^{N_{CP}}, \chi_{post}^{N_{CP}}, \rho_{post}^{N_{CP}}, \theta_{post}^{N_{CP}})\) after \(\text{edgeTime}\) clock ticks.

Upon the completion of Algorithm 3 using these values as parameters, \(\text{nextNode}\) contains marking \(\text{nextMarking}\), transition clocks \(\text{nextTransitionClocks}\), firing numbers \(\text{nextFiringNumbers}\) and global clock \(\text{nextGlobalClock}\) values such that \(\text{nextMarking} = m_{post}^{N_{CP}}, \text{nextTransitionClocks} = \chi_{post}^{N_{CP}}, \text{nextFiringNumbers} = \rho_{post}^{N_{CP}}\) and \(\text{nextGlobalClock} = \theta_{post}^{N_{CP}}\).

Proof. Follows from Lemmas 3-1, 3-2, 3-3, 3-4, 3-5, and 3-6. □

### 5.4 Algorithm 4

Algorithm \texttt{ConstructARGRecursive} is designed to build an augmented reachability graph \(RG_{CP}\) from a choice-point net \(N_{CP}\) a single node at a time. This approach is accomplished through recursion, where any newly-discovered nodes are subjected to a separate call to Algorithm 4 from within Algorithm 4 itself.

The current state of \(N_{CP}\) is assumed to be captured by the parameter \(\text{currentNode}\) with marking \(m\), transition clocks \(\chi\), firing numbers \(\rho\) and global clock \(\theta\). Parameter \(\text{nextTransitionsToFire}\) is a (possibly empty) set of transitions that is currently firing; Algorithm 4 updates this set if it has no members, \(i.e.,\) nothing is set to occur.

The immediate goal of this algorithm is to generate all possible subsequent nodes \(\text{nextNode}\) for \(\text{currentNode}\). These new nodes must correspond to the state of the system after the firing of an enabled transition (\(i.e.,\) one in \(\text{nextTransitionsToFire}\)) and the selection of a choice. Algorithm 4 tests all possible sequences of transitions in \(\text{nextTransitionsToFire}\) and all possible choices for those transitions. Any nodes which
have not been previously explored (i.e., those not already in the set of nodes \(X\) in \(RG_{CP}\)) are then subjected to a recursive call.

The first concern of Algorithm 4 is to test and update the set of transitions currently firing and the time elapsed between the firing of transitions. The following lemma formalizes the results of the updating process.

**Lemma 4-1.** Consider a choice-point net \(N_{CP}\) in state \((m, \chi, \rho, \theta)\) and let \(\text{currentNode} = (m, \chi, \rho, \theta)\). If the value of \(\text{nextTransitionsToFire}\) on line 1 is the empty set, then upon the completion of line 6 of Algorithm 4 \(\text{nextTransitionsToFire}\) is updated such that:

(i) for all \(t \in \text{nextTransitionsToFire}\), \(ED(t, m) > 0\)

(ii) for all \(t \in \text{nextTransitionsToFire}\), \(\rho(t) > 0\)

(iii) for all \(t_{sl} \in (\text{nextTransitionsToFire} \cap T_{SL})\) and for all \(u_{sl} \in (T_{SL} - \text{nextTransitionsToFire})\) where \(\chi(u_{sl}) \neq -1, \chi(t_{sl}) < \chi(u_{sl})\)

(iv) for all \(t_r \in (\text{nextTransitionsToFire} \cap T_{R})\) and for all \(u_r \in (T_{R} - \text{nextTransitionsToFire})\) where \(\chi(u_r) \neq -1, (\chi(t_r) - \theta) < (\chi(u_r) - \theta)\)

(v) for all \(t_{ml} \in (\text{nextTransitionsToFire} \cap T_{ML})\) and for all \(u_{ml} \in (T_{ML} - \text{nextTransitionsToFire})\), there exists \(x \in \chi(t_{sl})\) such that for all \(y \in \chi(u_{sl})\), \(x < y\)

(vi) for all \(v \in \text{nextTransitionsToFire}\), if \(v \in T_{SL}\) then \(\chi(v) = \text{edgeTime}\), if \(v \in T_{R}\) then \(\chi(v) - \theta = \text{edgeTime}\) and if \(v \in T_{ML}\) then \(\text{edgeTime} \in \chi(v)\)

**Proof.** The if statement on lines 1–3 contains the actions that produce the aforementioned result. Line 2 is responsible for determining the next transitions to fire according to conditions (i), (ii), (iii), (iv) and (v); a helper method \(\text{FindNextToFire}\)
Chapter 5. Behavioural Equivalence

examines each transition in $N_{CP}$ using the given requirements. Once nextTransitionsToFire has been updated, line 3 calls a second helper method GetTimeElapsed which calculates the time that must elapse according to condition (vi) and assigns it to edgeTime.

Conversely to Lemma 4-1, if a conflict is currently playing out (i.e., nextTransitionsToFire $\neq \emptyset$) then the time elapsed between the firing of transitions must be set to zero. Transitions in conflict occur in near-simultaneous succession; any edges resulting from these events are conflict edges and therefore associated with zero time (which is assigned to edgeTime on line 5).

As stated previously, Algorithm 4 is responsible for generating all possible subsequent nodes nextNode for given node currentNode. These nodes must accurately capture the state of the choice-point net after an enabled transition fires and a choice is selected. Therefore, a node nextNode must be generated for each of the enabled transitions and their choices. The following lemma demonstrates that this is the case.

**Lemma 4-2.** Let $currentNode = (m, \chi, \rho, \theta)$ represent a reachable state for choice-point net $N_{CP}$ where transitions nextTransitionsToFire are currently firing. Let $t \in nextTransitionsToFire$ and let $c \in C_T(t)$. Following the firing of $t$ and the selection of $c$, let $N_{CP}$ be in state $(m^N_{CP}, \chi^N_{CP}, \rho^N_{CP}, \theta^N_{CP})$. Upon the conclusion of Algorithm 4, a node nextNode $= (m^N_{post}, \chi^N_{post}, \rho^N_{post}, \theta^N_{post})$ is generated for each enabled transition $t$ and associated choice $c$.

**Proof.** Every transition in nextTransitionsToFire is considered in lines 7–19; the same can be said for choices in lines 8–18. The next node in the augmented reachability graph is determined on line 10 by calling Algorithm 3. Note that the assumptions for Proposition 5 are satisfied by Lemma 4-1, the forall statements on lines 7 and
8, the use of \texttt{recNextTransitionsToFire} (copied from \texttt{nextTransitionsToFire} on line 9) in the call to Algorithm 3 on line 10, and the recursive call to Algorithm 4 on line 16 with \texttt{nextNode} and \texttt{recNextTransitionsToFire}. Therefore, \texttt{nextNode} is representative of the next state of the $N_{CP}$.

Although it is vital that each node created contains a reachable state for the choice-point net, it is equally important that the edges connecting these nodes capture what occurred to cause the change in state. The following lemma does for edges what Lemma 4-2 does for nodes.

\textbf{Lemma 4-3.} Let $\texttt{currentNode} = (m, \chi, \rho, \theta)$ represent a reachable state for choice-point net $N_{CP}$ where transitions $\texttt{nextTransitionsToFire}$ are currently firing. Let $t \in \texttt{nextTransitionsToFire}$ and let $c \in C_T(t)$. Let $\texttt{nextNode}$ represent the state of $N_{CP}$ following the firing of $t$ and the selection of $c$, which occurred with probability $q$ after $k$ clock ticks. Upon the conclusion of Algorithm 4, an edge $\texttt{nextEdge} = (t, c, \texttt{nextProbability}, \texttt{edgeTime})$ connects $\texttt{currentNode}$ to $\texttt{nextNode}$ in $RG_{CP}$ where $\texttt{nextProbability} = q$ and $\texttt{edgeTime} = k$.

\textit{Proof.} It was established in Lemma 4-2 that each possible transition and choice are examined and a node is generated. The probability of the choice $c$ is normalized on line 11 (if necessary) using $\texttt{nextTransitionsToFire}$, followed by the creation of the edge on line 12 with the choice’s label, the probability and the time. Note that the time $\texttt{edgeTime}$ was established to be correct if $\texttt{nextTransitionsToFire} = \emptyset$ in Lemma 4-1. If $\texttt{nextTransitionsToFire} \neq \emptyset$, lines 4–6 are executed and line 5 is responsible for assigning a value of zero to $\texttt{edgeTime}$. The transition function is then updated on line 13 to reflect the connection between $\texttt{currentNode}$ and $\texttt{nextNode}$. \hfill $\Box$
The previous lemmas have established the functionality of Algorithm 4—all that remains is to connect and formalize these properties into a single proposition.

**Proposition 6.** Consider a choice-point net $N_{CP}$ in state $(m, \chi, \rho, \theta) \in \Psi$, which is represented by parameter $\text{currentNode}$. Let $\text{nextTransitionsToFire} = \emptyset$ or $\text{nextTransitionsToFire} \subseteq T$ where these transitions may fire in near-simultaneous succession. For all $t \in \text{nextTransitionsToFire}$ and all $c \in C_T(t)$, following the firing of $t$ and the selection of $c$ let $N_{CP}$ be in state $(m_{\text{post}}^{N_{CP}}, \chi_{\text{post}}^{N_{CP}}, \rho_{\text{post}}^{N_{CP}}, \theta_{\text{post}}^{N_{CP}})$. Assume that $t$ and $c$ have occurred with probability $q$ after $k$ clock ticks. Upon the completion of Algorithm 4, a node $\text{nextNode} = (m_{\text{post}}^{N_{CP}}, \chi_{\text{post}}^{N_{CP}}, \rho_{\text{post}}^{N_{CP}}, \theta_{\text{post}}^{N_{CP}})$ has been generated and added to the augmented reachability graph’s set of nodes $X$, assuming it is not currently present. An edge $\text{nextEdge} = (t, c, q, k)$ has been created and added to $RG_{CP}$ such that $\delta(\text{currentNode}, \text{nextEdge}) = \text{nextNode}$. Finally, Algorithm 4 is called recursively on all nodes $\text{nextNode}$ that have not be previously discovered (i.e., $\text{nextNode} \notin X$) with $\text{recNextTransitionsToFire}$.

**Proof.** The majority of this proof follows from Lemmas 4-1, 4-2 and 4-3. The remainder rests on lines 14–17 which begin by checking whether $\text{nextNode}$ is already present in $X$. If not, $\text{nextNode}$ is added to the set on line 15 and a recursive call to Algorithm 4 is made on line 16.

**Corollary 1.** Consider a choice-point net $N_{CP}$ in state $(m, \chi, \rho, \theta) \in \Psi$, which is represented by parameter $\text{currentNode}$. Let $\text{nextTransitionsToFire}$ represent the set of transitions that are currently firing. Upon the conclusion of Algorithm 4, every new node $\text{nextNode}$ and every new edge $\text{nextEdge}$ are generated as the result of the firing of
an enabled transition \( t \in \text{nextTransitionsToFire} \) and the selection of a choice \( c \in C_T(t) \).
No nodes or edges are created for any other reason.

Proof. Lemmas 4-2 and 4-3 establish that the nodes and edges are generated in this manner. This occurs in the \textbf{forall} loops on lines 7–19 and 8–18. Algorithm 4 does not create nodes or edges at any other point in the procedure.

5.5 Algorithm 5

The purpose of \texttt{ConstructARG} is to begin the construction of \( \mathcal{RG}_{CP} \) by creating its initial form (i.e., simply containing its initial node) and calling Algorithm 4 to recursively complete the graph. As such, Algorithm 5 is remarkably simple; it comprises just nine lines. However, it is also the algorithm that creates an augmented reachability graph \( \mathcal{RG}_{CP} \) from a choice-point net \( N_{CP} \) and an initial global clock value \( GCV \). Thus, Algorithm 5 is associated with the following essential theorem which proves that \( \mathcal{RG}_{CP} \) does preserve the behaviour of \( N_{CP} \).

**Theorem 1.** Given a choice-point net \( N_{CP} = (P,T,I,C,R,SL,ML,S,MD,CU,M,m_0) \), let \( \Sigma_{GCV,m_0} = \{\sigma_0, \sigma_1, \ldots\} \) contain all possible firings starting with initial clock value \( GCV \) and initial marking \( m_0 \). Let \( \sigma_i = \psi_i^0|\omega_i^1|\cdots|\omega_i^j|\psi_i^j \) where \( \omega_i^k = (t_i^k, c_i^k, p_i^k, k_i^k) \) represents the event and \( \psi_i^k = (m_i^k, \pi_i^k, \rho_i^k, \theta_i^k) \) represents the state of the choice-point net for \( i, j \in \mathbb{N} \) and \( 0 \leq k \leq j \). The length of \( \sigma_i \), denoted by \( |\sigma_i| \), is defined to be the number of events in \( \sigma_i \), i.e., for \( \sigma_i = \psi_i^0|\omega_i^1|\cdots|\omega_i^j|\psi_i^j \), \(|\sigma_i| = j\).

Using \( N_{CP} \) and \( GCV \) as arguments, Algorithm 5 will return an associated augmented reachability graph \( \mathcal{RG}_{CP} = (\Omega, X, x_0) \). Let \( P_{GCV,m_0} = \{p_0, p_1, \ldots\} \) contain all possible paths \( p_a = x_a^0 \xrightarrow{e_a^1} x_a^1 \cdots \xrightarrow{e_a^b} x_a^b \) starting from the initial node of the graph.
Let \( x_a^d = (m_a^d, \chi_a^d, \rho_a^d, \theta_a^d) \) represent a node in the graph and \( e_a^d = (t_a^d, c_a^d, q_a^d, k_a^d) \) represent an edge in the graph for \( a, b_a \in \mathbb{N} \) and \( 0 \leq d \leq b_a \). The length of \( p_a \), denoted by \(|p_a|\), is defined to be the number of edges in \( p_a \), i.e., for \( p_a = x_a^0 \xrightarrow{e_1^a} x_a^1 \cdots \xrightarrow{e_{b_a}} x_a^{b_a} \), \(|p_a| = b_a\).

For all firings \( \sigma_i \), there exists a path \( p_i \) such that \(|\sigma_i| = |p_i| = r\) and for all \( s \) such that \( 0 \leq s \leq r \), \( \psi_i^s = x_i^s \) and \( \omega_i^s = e_i^s \). Conversely, for all paths \( p_i \) there exists a firing \( \sigma_i \) such that \(|p_i| = |\sigma_i| = r\) and for all \( s \) such that \( 0 \leq s \leq r \), \( x_i^s = \psi_i^s \) and \( e_i^s = \omega_i^s \), i.e., the returned augmented reachability graph \( \mathcal{RG}_{CP} \) preserves the behaviour captured by the given choice-point net \( \mathcal{N}_{CP} \).

(Proof) Each \( \sigma_i \) has a corresponding \( p_i \). **Base Case:** \( \sigma_0 \) where \(|\sigma_0| = 0\)

Let \( \sigma_0 = \psi_0^0 \) where \( \psi_0^0 = (m_0, \chi_0^0, \rho_0^0, GCV) \), i.e., no events have occurred. When Algorithm 5 is called both the initial marking \( m_0 \) and initial global clock value \( GCV \) are assigned to \( \text{initialNode} \) on line 4. With respect to the transition clock values, a new clock function \( \chi \) is created with entries of \(-1\) or \( \emptyset \) on line 1 and is updated using Algorithm 1 on line 5. The assumptions of Proposition 2 are satisfied: all entries in \( \chi \) for non-multiple lapse transitions are \(-1\) and therefore, after Algorithm 1 is called and has returned, \( \chi_0^0 = \chi \). Similarly, a new firing number function \( \rho \) is created with entries of 0 on line 2 and is updated using Algorithm 2 on line 6. The assumptions of Proposition 4 are satisfied: all entries in \( \rho \) are 0 and therefore, after Algorithm 2 is called and has returned, \( \rho_0^0 = \rho \). The variable \( \text{initialNode} \) is designated as the initial node of \( \mathcal{RG}_{CP} \) on line 7. Let \( p_0 = x_0^0 = \text{initialNode} \). Therefore, \( \psi_0^0 = x_0^0 \) and \( \sigma_0 = p_0 \).

**Inductive Hypothesis:** For each firing \( \sigma_i \) where \(|\sigma_i| = n\), there exists a path \( p_j \) where \(|p_j| = n \) and \( m_i^l = m_j^l \), \( \theta_i^l = \theta_j^l \), \( \chi_i^l = \chi_j^l \), \( \rho_i^l = \rho_j^l \), \( t_i^l = t_j^l \), \( c_i^l = c_j^l \), \( q_i^l = q_j^l \) and

\( t_a^d = (m_a^d, \chi_a^d, \rho_a^d, \theta_a^d) \) represent a node in the graph and \( e_a^d = (t_a^d, c_a^d, q_a^d, k_a^d) \) represent an edge in the graph for \( a, b_a \in \mathbb{N} \) and \( 0 \leq d \leq b_a \). The length of \( p_a \), denoted by \(|p_a|\), is defined to be the number of edges in \( p_a \), i.e., for \( p_a = x_a^0 \xrightarrow{e_1^a} x_a^1 \cdots \xrightarrow{e_{b_a}} x_a^{b_a} \), \(|p_a| = b_a\).

For all firings \( \sigma_i \), there exists a path \( p_i \) such that \(|\sigma_i| = |p_i| = r\) and for all \( s \) such that \( 0 \leq s \leq r \), \( \psi_i^s = x_i^s \) and \( \omega_i^s = e_i^s \). Conversely, for all paths \( p_i \) there exists a firing \( \sigma_i \) such that \(|p_i| = |\sigma_i| = r\) and for all \( s \) such that \( 0 \leq s \leq r \), \( x_i^s = \psi_i^s \) and \( e_i^s = \omega_i^s \), i.e., the returned augmented reachability graph \( \mathcal{RG}_{CP} \) preserves the behaviour captured by the given choice-point net \( \mathcal{N}_{CP} \).

(Proof) Each \( \sigma_i \) has a corresponding \( p_i \). **Base Case:** \( \sigma_0 \) where \(|\sigma_0| = 0\)

Let \( \sigma_0 = \psi_0^0 \) where \( \psi_0^0 = (m_0, \chi_0^0, \rho_0^0, GCV) \), i.e., no events have occurred. When Algorithm 5 is called both the initial marking \( m_0 \) and initial global clock value \( GCV \) are assigned to \( \text{initialNode} \) on line 4. With respect to the transition clock values, a new clock function \( \chi \) is created with entries of \(-1\) or \( \emptyset \) on line 1 and is updated using Algorithm 1 on line 5. The assumptions of Proposition 2 are satisfied: all entries in \( \chi \) for non-multiple lapse transitions are \(-1\) and therefore, after Algorithm 1 is called and has returned, \( \chi_0^0 = \chi \). Similarly, a new firing number function \( \rho \) is created with entries of 0 on line 2 and is updated using Algorithm 2 on line 6. The assumptions of Proposition 4 are satisfied: all entries in \( \rho \) are 0 and therefore, after Algorithm 2 is called and has returned, \( \rho_0^0 = \rho \). The variable \( \text{initialNode} \) is designated as the initial node of \( \mathcal{RG}_{CP} \) on line 7. Let \( p_0 = x_0^0 = \text{initialNode} \). Therefore, \( \psi_0^0 = x_0^0 \) and \( \sigma_0 = p_0 \).

**Inductive Hypothesis:** For each firing \( \sigma_i \) where \(|\sigma_i| = n\), there exists a path \( p_j \) where \(|p_j| = n \) and \( m_i^l = m_j^l \), \( \theta_i^l = \theta_j^l \), \( \chi_i^l = \chi_j^l \), \( \rho_i^l = \rho_j^l \), \( t_i^l = t_j^l \), \( c_i^l = c_j^l \), \( q_i^l = q_j^l \) and
\( k^l_i = k^l_j \) for \( 1 \leq l \leq n \).

**Inductive Step:** \( |\sigma_q| = n + 1 \)

Let \( \sigma_i \) represent the first \( n \) steps of the firing \( \sigma_q \), i.e., \( \sigma_q = \sigma_i | \omega^{n+1}_q \psi^{n+1}_q \). Let \( \omega^{n+1}_q = (t, c, p, k) \) represent the event that took \( \sigma_i \) to \( \psi^{n+1}_q \). From the inductive hypothesis, there exists a path \( p_j \) such that \( p_j = \sigma_i \). What is required is to prove there exists an edge \( e^{n+1}_r = (t, c, q, k) \) from \( p_j \) to node \( x^{n+1}_r \) where \( x^{n+1}_r = \psi^{n+1}_q \).

Let \( x^n_j \) be the last node in path \( p_j \); it will have been generated by a recursive call to Algorithm 4 during the construction of \( p_j \). When \( x^n_j \) is first created in Algorithm 4, Algorithm 4 is then called recursively (on line 16) with \( x^n_j \) as parameter \texttt{currentNode}. After transition \( t \) fires and choice \( c \) is made (according to the firing \( \sigma_q \)), Proposition 6 requires that a node \texttt{nextNode} \( (x^{n+1}_r) \) be generated for this possibility and connected to the \texttt{currentNode} \( (x^n_j) \) via \texttt{nextEdge} \( (e^{n+1}_r) \). Proposition 6 also requires that \( x^{n+1}_r \) accurately capture that state of \( N_{CP} \) after the event occurs, i.e., \( x^{n+1}_r = \psi^{n+1}_q \). Finally, Proposition 6 ensures that \( \omega^{n+1}_q = e^{n+1}_r \).

(Proof) Each \( p_i \) has a corresponding \( \sigma_i \). **Base Case:** \( p_0 \) where \( |p_0| = 0 \)

Let \( p_0 = x^0_0 \) where \( x^0_0 = (m_0, \chi, \rho, GCV) \), i.e., no events have occurred. The path \( p_0 \) consists of the initial state of \( R \mathcal{G}_{CP} \) and is created in Algorithm 5 on lines 1–4. The empty firing \( \sigma_0 = \psi^0_0 = (m_0, \chi^0_0, \rho^0_0, GCV) \) consists of the initial state of \( N_{CP} \). The path \( p_0 \) was constructed using identical elements from \( \sigma_0 \) with the exception of \( \chi^0_0 \) and \( \rho^0_0 \). However, the assumptions of Propositions 2 and 4 are satisfied and therefore \( \chi = \chi^0_0 \) following the call to Algorithm 1 while \( \rho = \rho^0_0 \) following the call to Algorithm 2. Therefore, \( x^0_0 = \psi^0_0 \) and \( p_0 = \sigma_0 \).

**Inductive Hypothesis:** For each path \( p_j \) where \( |p_j| = n \), there exists a firing \( \sigma_i \)
where $|\sigma_i| = n$ such that $m^l_j = m^l_i$, $\theta^l_j = \theta^l_i$, $\chi^l_j = \chi^l_i$, $\rho^l_j = \rho^l_i$, $t^l_j = t^l_i$, $c^l_j = c^l_i$, $q^l_j = q^l_i$ and $k^l_j = k^l_i$ for $1 \leq l \leq n$.

**Inductive Step:** $|p_r| = n + 1$

Let $p_j$ represent the first $n$ steps of the path $p_r$, i.e., $p_r = p_j \xrightarrow{e^{n+1}_r} x^{n+1}_r$. Let $e^{n+1}_r = (t, c, q, k)$ represent the edge connecting $p_j$ to $x^{n+1}_r$. From the inductive hypothesis, there exists a firing $\sigma_i$ such that $p_j = \sigma_i$. What is required is to prove there exists an event $\omega^{n+1}_q = (t, c, p, k)$ from firing $\sigma_i$ to state $\psi^{n+1}_q$ where $\psi^{n+1}_q = x^{n+1}_r$.

Let $x^n_j$ be the last node in path $p_j$; it will have been generated by a recursive call to Algorithm 4 during the construction of $p_j$. Corollary 1 guarantees that each nextNode ($x^{n+1}_r$) and nextEdge ($e^{n+1}_r$) generated represents the result of firing an enabled transition $t$ and selecting an associated choice $c$. If $t$ is allowed to fire under the timing constraints and $c$ is a valid choice then a firing of $N_{CP}$ must exist where $\sigma_i|\omega^{n+1}_q \psi^{n+1}_q$ such that $\omega^{n+1}_q = e^{n+1}_r$ and $\psi^{n+1}_q = x^{n+1}_r$. Therefore, there exists firing $\sigma_q = \sigma_i|\omega^{n+1}_q \psi^{n+1}_q$ such that $\sigma_q = p_r$. □
Chapter 6

Question Restrictions

Although choice-point nets are an appropriate tool for modelling health-care protocols (and perhaps other systems), a model alone will not answer an administrator’s questions about a set of guidelines. The ultimate goal of this research is to provide a framework for analysis that can be customized to answer a wide variety of questions posed. In this manner, choice-point nets may prove useful to the health-care community, rather than simply interesting from a research perspective.

Consider a health-care protocol that has been converted into a mathematical model, \( N_{CP} \), and its associated augmented reachability graph \( RG_{CP} \). A great deal of information about the evolution of \( N_{CP} \) is stored in the nodes and edges of \( RG_{CP} \)—all possible sequences of states and events, each of which contains marking, timing and probability data. In order to answer a person’s questions about the protocol, his or her prose questions must be translated into a language suitable for the graph.

The solution lies in considering these queries from another perspective. Within the augmented reachability graph, certain paths (starting from the initial node) will
satisfy or “answer” a question while other paths will not. For example, the query “Can the entire population become ill?” is solely interested in paths that lead to nodes with markings which represent that outcome. Similarly, the question “What is the chance that more than ten failures occur” is only concerned with paths whose edges are labelled with at least ten failure events or choices. From this perspective, healthcare questions may be seen as restrictions on the exploration of $\mathcal{RG}_{CP}$ according to desirable or undesirable outcomes of interest. More detailed queries may be composed of several restrictions. For example, “What are the chances that over 50% of the population will become ill within 72 hours?” places limits on both time and markings.

Each prose question may be represented by a set of question restrictions $QR$ that provide a stopping point when searching for paths in the graph, i.e., further exploration is restricted once the conditions are met. At the core of a single question restriction $qr \in QR$ is a comparison (e.g., “more than”, “within”, “equal to”).

Mathematically, these may be expressed as an operator from the set $O = \{eq, lt, leq, gt, geq\}$. The set $O$ consists of standard comparator operations, and an element $o$ of this set will be applied in one of several contexts. If probabilities are being compared, the operator $o$ will compare two numbers from $[0, 1]$ and produce a binary outcome. If considering time, $o$ will compare two natural numbers and produce a binary outcome. The appropriate domain for the operator will be understood from the context of the restriction (e.g., probability or time). The operators in $O$ are defined as follows:

- $eq(a, b) := (a == b)$
- $lt(a, b) := (a < b)$
- $leq(a, b) := (a \leq b)$
• $gt(a, b) := (a > b)$

• $geq(a, b) := (a \geq b)$

The purpose of these operators is to define how the data stored within $\mathcal{RG}_{CP}$ will be compared to user-defined values (e.g., “24 hours”). However, each restriction must also specify what type of data is under consideration: markings, probability, time or choices. This is accomplished by defining each question restriction $qr \in QR$ as a member of one of the following sets:

• marking restrictions $MR \subseteq P \times O \times N$ where $(p, o, i) \in MR$ requires $o(M(p), i)$ to be true

• probability restrictions $PR \subseteq \{\text{prob}\} \times O \times [0, 1]$ where $(\text{prob}, o, b) \in PR$ requires $o(l, b)$ to be true such that $l$ is the cumulative probability of a possible path

• time restrictions $TR \subseteq \{\text{time}\} \times O \times N$ where $(\text{time}, o, u) \in TR$ requires $o(l, u)$ to be true such that $l$ is the cumulative time of a possible path

• choice restrictions $CR \subseteq C \times O \times N$ where $(d, o, i) \in CR$ requires $o(e, i)$ to be true such that $e$ is the total number of occurrences of the choice $d$ on a possible path

Consider a single question restriction $r$ belonging to one of the aforementioned sets. Assuming there exist some paths or strings in $\mathcal{RG}_{CP}$ that satisfy $r$, let $\mathcal{L}(\mathcal{RG}_{CP})_r$ represent the language formed by this set of strings or runs. Let $r$ and $s$ be two restrictions; when applied jointly, both $r$ and $s$ can act as a single restriction. Formally, $r \land s$ represents the conjunction of the associated predicates. A path must satisfy
each restriction individually in order to satisfy them jointly. Therefore,

$$L(RG_{CP})_{rs} = L(RG_{CP}) r \cap L(RG_{CP}) s$$

A similar assertion may be made for $L(RG_{CP})_{rs}$ using the disjunctive operator $\lor$ and union operator $\cup$.

These definitions may be extended to any possible combination of question restrictions contained in set $QR$. There are two approaches to applying this set of restrictions: with conjunction or disjunction. Formally, $QR = qr_1 \land \ldots \land qr_m$ where $(qr_i \in MR) \lor (qr_i \in PR) \lor (qr_i \in TR) \lor (qr_i \in CR)$ for all $1 \leq i \leq m$. The sublanguage $L(RG_{CP})_{QR} \subseteq L(RG_{CP})$ is the set of paths or runs defined as follows:

$$L(RG_{CP})_{QR} = L(RG_{CP})_{qr_1 \land \ldots \land qr_m} = L(RG_{CP})_{qr_1} \cap \ldots \cap L(RG_{CP})_{qr_m}$$

Again, a similar assertion may be made for $L(RG_{CP})_{QR}$ with the disjunctive operator $\lor$.

6.1 Applying Restrictions

A question about a health-care protocol is typically defined according to requirements that must be satisfied (e.g., “less than 24 hours”), conditions that cannot be violated (e.g., “no more than two failures”), or a combination thereof. When asking a question, health-care administrators are essentially placing limits on which sequences of events generated by the model are of interest and which are not. Within an augmented reachability graph $RG_{CP}$, a path may meet the conditions or violate them. Applying
a set of restrictions essentially boils down to finding satisfactory runs within the reachability graph and discarding the rest.

Question restrictions are divided into two sets: an acceptance set $A$ and a rejection set $J$. A given path must satisfy each condition in the acceptance set to be considered part of the answer. In contrast, a path that violates any condition in the rejection set is immediately discarded. Acceptance restrictions are evaluated using conjunction to ensure all requirements are met while rejection restrictions are evaluated using disjunction to ensure that nothing undesirable is permitted. Formally, the languages generated by $A$ and $J$ are $\mathcal{L}(\mathcal{RG}_{CP})_A$ and $\mathcal{L}(\mathcal{RG}_{CP})_J$, respectively.

Consider the following question in the context of the running LTCH example: what is the probability that both residents’ illness will be discovered within 24 hours? Satisfactory runs must see the event $sSeeIllResident$ and choice $\text{Isolate}$ occur twice while sequences that occur over more than 24 hours are not valid. This yields an acceptance set of $A = \{(\text{Isolate}, \geq, 2)\}$ and a rejection set of $J = \{(\text{time}, \gt, 24)\}$. A second question concerns the likelihood that both residents are still undiscovered after 48 hours, resulting in $A = \{(\text{rIll}, \neq, 2), (\text{time}, \geq, 48)\}$ and $J = \{(\text{Isolate}, \gt, 0)\}$.

Depending on the layout of the augmented reachability graph, it is possible that exploration under a set of question restrictions may not terminate, i.e., a loop in the graph provides an infinite path that cannot be accepted or rejected. It is therefore necessary to add a restriction to the rejection set that caps the exploration of any given path. Since both probability and time are common to all paths (unlike specific choices or markings), specifying a large time value or a small probability value as a restriction (e.g., $J = \{(\text{time}, \gt, 10^3), (\text{prob}, \lt, 10^{-6})\}$) would guarantee termination. Although some valid paths may be discarded using this approach, the probability of
any path decreases as it grows in length; discarding a lengthy non-terminating path with an extremely low probability is highly unlikely to skew the results.

For example, consider the acceptance set $A = \{ (\text{Ignore}, gt, 1) \}$—it determines the chances that any resident’s symptoms will be ignored more than once. However, there are two loops in Figure 4.2 composed strictly of $\text{sSeeIllResident}$ events and $\text{Ignore}$ choices, which will both result in non-terminating paths. This necessitates the addition of a rejection set $J = \{ (\text{prob}, lt, 10^{-6}) \}$ to prevent the algorithms from entering an infinite loop.

### 6.2 Exploration Algorithms

To determine the answer to a given question, the associated acceptance and rejection sets can be passed into Algorithm 6 along with the augmented reachability graph $\mathcal{RG}_{CP}$. This algorithm’s only task is to begin the recursive exploration; this is accomplished by giving the initial node of the augmented reachability graph to Algorithm 7 as the starting point and an initial path probability of 1.0 as $\mathcal{N}_{CP}$ always begins in its initial state. Algorithm 6 begins with the base case for the recursive call, thus eliminating the need for someone to determine it in order to get an answer to the question.

\begin{algorithm}
\caption{AnswerQuestion}
\textbf{Input}: An augmented reachability graph $\mathcal{RG}_{CP} = (\Omega, X, \delta, x_0)$
\hspace{1cm} The set of acceptance restrictions $A$
\hspace{1cm} The set of rejection restrictions $J$
\textbf{Output}: The total probability of all paths that satisfy the given restrictions

\begin{algorithmic}
1. \textbf{return} AnswerQuestionRecursive($\mathcal{RG}_{CP}$, $x_0$, $A$, $J$, 1.0)
\end{algorithmic}
\end{algorithm}
Algorithm 7: AnswerQuestionRecursive

Input: An augmented reachability graph $RG_{CP} = (\Omega, X, \delta, x_0)$
The current node $currentNode = (m, \chi, \rho, \theta)$
The set of acceptance restrictions $A$
The set of rejection restrictions $J$
The probability of the current path $pathProbability$

Output: The total probability of all paths that satisfy the given restrictions

1) $totalProbability \leftarrow 0.0$
2) $forall the nextEdge = (t, c, q, k) where \delta(currentNode, nextEdge) \neq \bot do$
3) $nextNode \leftarrow \delta(currentNode, nextEdge)$
   // Check whether the restrictions have been met
4) if SatisfiesRestrictions(nextNode, nextEdge, $J$, $\lor$) then
   5) continue
5) else if SatisfiesRestrictions(nextNode, nextEdge, $A$, $\land$) then
   6) // If the next node is an accept, add its
   7) // path probability
   8) $totalProbability \leftarrow totalProbability + (pathProbability \times q)$
else
9) // If the next node is neither accepted or
10) // rejected, go through each outgoing edge and
   // prepare to explore further
   $nextAccept \leftarrow copy of A$
   $nextReject \leftarrow copy of J$
   // Alter the restrictions to reflect what
   // occurred in this part of the recursion
11) AdjustRestrictions(nextNode, nextEdge, nextAccept)
12) AdjustRestrictions(nextNode, nextEdge, nextReject)
13) $totalProbability \leftarrow totalProbability + AnswerQuestionRecursive(RG_{CP}, nextNode,
    nextAccept, nextReject, pathProbability \times q)$
end
14) end
15) return $totalProbability$

The first task in Algorithm 7 is to determine whether the current path (ending with the next node) satisfies one of the rejection restrictions or all of the acceptance restrictions. In the latter case, the probability of the given path is added; in the former, nothing is recorded as the path is deemed invalid. If neither set of restrictions is satisfied, a recursive search must take place on all outgoing edges.

Rather than maintain separate parameters to track the properties of the current
path (e.g., cumulative number of choices, total time elapsed), the algorithms maintain this data in a different manner. Before a recursive call is made, new sets of question restrictions are generated from the current set and “adjusted” to reflect what has occurred during this latest step in the graph. For example, a time restriction \((time, \leq, 24)\) may be adjusted to \((time, \leq, 20)\) following an edge with a time of four hours without altering the results of the exploration.

Consider a run \(r = x_0 \xrightarrow{e_1} x_1 \ldots \xrightarrow{e_n} x_n\) in \(\mathcal{RG}_{CP}\) and a question restriction \(qr \in QR\). Let \(e_k = (t_k, c_k, q_k, k_k)\) and \(x_k = (m_k, \chi_k, \rho_k, \theta_k)\) for all \(1 \leq k \leq n\). For a probability restriction \(qr = (prob, o, b)\) where the cumulative probability \((q_1 \times q_2 \times q_3 \times \ldots \times q_n)\) must be tracked,

\[
o(q_1 \times q_2 \times q_3 \times \ldots \times q_n, b) = o(q_2 \times q_3 \times \ldots \times q_n, b \div q_1) = o(q_3 \times \ldots \times q_n, (b \div q_1) \div q_2) = \ldots = o(q_n, ((b \div q_1) \div q_2) \div \ldots \div q_{n-1})
\]

For example, let \(r_{e.g.} = x_0 \xrightarrow{(t_1, c_1, 0.2, 8)} x_1 \xrightarrow{(t_1, c_2, 0.5, 0)} x_2 \xrightarrow{(t_2, c_3, 0.4, 15)} x_3 \xrightarrow{(t_1, c_1, 0.2, 0)} x_4\). For the initial probability restriction \((prob, lt, 0.01)\) to be satisfied:

\[
0.2 \times 0.5 \times 0.4 \times 0.2 \ < \ 0.01 \\
0.5 \times 0.4 \times 0.2 \ < \ 0.05 \\
0.4 \times 0.2 \ < \ 0.1 \\
0.2 \ < \ 0.25
\]

The run \(r_{e.g.}\) satisfies the given probability restriction. For a time restriction \(qr = \)
(time, o, u) where the total time elapsed \((k_1 + k_2 + k_3 + \ldots + k_n)\) is under consideration,

\[
o(k_1 + k_2 + k_3 + \ldots + k_n, u) = o(k_2 + k_3 + \ldots + k_n, u - k_1) = o(k_3 + \ldots + k_n, u - k_1 - k_2) = \ldots = o(k_n, u - k_1 - k_2 + k_3 - \ldots - k_{n-1})
\]

Given an initial time restriction \((time, gt, 24)\) and the run \(r_{e.g.}\):

\[
8 + 0 + 15 + 0 > 24 \\
0 + 15 + 0 > 16 \\
15 + 0 > 16 \\
0 > 1
\]

In this case, \(r_{e.g.}\) does not yet satisfy the given time restriction. Finally, for a choice restriction \(qr = (d, o, i)\) which is concerned with the total number of edges in \(r\) with choice \(d\) (represented here by \(d_r\)),

\[
o(d_r, i) = o(\underbrace{1 + 1 + 1 + \ldots + 1}_d, i) = o(\underbrace{1 + 1 + \ldots + 1}_{d_r - 1}, i - 1) = o(\underbrace{1 + \ldots + 1}_{d_r - 2}, i - 1) = \ldots = o(\underbrace{1, i - 1 - 1 - \ldots - 1}_{d_r - 1})
\]
For example, consider an initial choice restriction \((c_1, leq, 2)\) and the run \(r_{e.g.}\):

\[
\begin{align*}
1 + 0 + 0 + 1 & \leq 2 \\
0 + 0 + 1 & \leq 1 \\
0 + 1 & \leq 1 \\
1 & \leq 1
\end{align*}
\]

The run \(r_{e.g.}\) does satisfy the given choice restriction.

Algorithm 8 is responsible for performing these adjustments. This method en-

---

**Algorithm 8: AdjustRestrictions**

**Input:** A current node \(currentNode = (m, \chi, \rho, \theta)\)
A current edge \(currentEdge = (t, c, q, k)\)
A set of question restrictions \(QR\)

1) **forall the** \(qr \in QR\) **do**
2) **if** \(qr \in PR \subseteq \{\text{prob}\} \times O \times [0, 1] \) where \(qr = (\text{prob}, o, b)\) **then**
   3) \(qr \leftarrow (\text{prob}, o, b \div q)\)
4) **else if** \(qr \in TR \subseteq \{\text{time}\} \times O \times N\) where \(qr = (\text{time}, o, u)\) **then**
   5) \(qr \leftarrow (\text{time}, o, u - k)\)
6) **else if** \(qr \in CR \subseteq C \times O \times N\) where \(qr = (d, o, i)\) **then**
   7) **if** \(d = c\) **then**
   8) \(qr \leftarrow (d, o, i - 1)\)
9) **end**
10) **end**
11) **end**

---

forces the aforementioned strategies: marking restrictions are not adjusted as they must remain constant; probability restrictions have their values divided by the probability attached to the current edge; timing restrictions are reduced by the time attached to the current edge; and choice restrictions are reduced by one if the choice attached to the current edge is identical.

Algorithm 9 is responsible for testing the current path against the current set of restrictions—it is called from within Algorithm 7. This method is straightfor-
ward with the possible exception of how choice restrictions are handled. Since an
edge represents the selection of a single choice, the number of choices given in the
restriction is compared to one. This approach works thanks to the restriction adjust-
ments performed in Algorithm 8—because the desired number of choice occurrences
is decremented each time that choice appears, the restriction is satisfied whenever a
single occurrence is left and the current choice matches it.

Returning to the LTCH example, these four algorithms provide the means to
answer the questions posed earlier in this chapter. The first set of restrictions
$A = \{(\text{Isolate}, \geq, 2)\}$ and $J = \{(\text{time}, \gt, 24)\}$ yields three paths: $(sSIR, \text{Isolate}) - (sSIR, \text{Isolate}), (sSIR, \text{Ignore}) - (sSIR, \text{Isolate}) - (sSIR, \text{Isolate})$ and $(sSIR, \text{Isolate}) - (sSIR,$
Ignore)\text{—}(sSIR, Isolate). The cumulative probabilities for these paths are $(0.8 \cdot 0.8)$, $(0.2 \cdot 0.8 \cdot 0.8)$ and $(0.8 \cdot 0.2 \cdot 0.8)$. This implies that the chances of both residents being isolated within 24 hours is $0.64 + 0.128 + 0.128 = 0.896$ or 89.6%.

The second set of restrictions $A = \{(r11, eq, 2), (time, geq, 48)\}$ and $J = \{(Isolate, gt, 0)\}$ yields a single path $(sSIR, Ignore)\text{—}(sSIR, Ignore)\text{—}(sSIR, Ignore)\text{—}(sSIR, Ignore)$ with a probability of $0.2^{10} = 1.024 \cdot 10^{-7}$. This means that the likelihood of neither residents’ illness being discovered after 48 hours is 0.00001024%.

### 6.3 Question Restrictions and Model Checking

Searching the paths of an augmented reachability graph for certain properties may appear to be similar to model checking [22], [21], [27], but the analysis performed by these methods is different. Given a structure $M$ and a logical formula $\phi$, a model checker will decide if $M$ satisfies $\phi$, denoted by $M \models \phi$. The logical formula $\phi$ is defined over a set of propositions and is often expressed using a temporal logic such as computation tree logic (CTL), linear temporal logic (LTL) and propositional linear-time logic (PLTL). The temporal logic employed is often customized to manage the particular properties of the structure $M$ [44], [47], [32].

Using a model checker it is possible to determine whether $M \models \phi$. However, it is not possible to produce the same results as question restrictions for two reasons:

1. Evaluating whether a structure $M$ is a model of a logical formula $\phi$ will yield one of two results: yes or no (possibly with a counterexample). There is no approach that returns the probability that a particular formula is satisfied, as is the case with question restrictions.
2. Model checking considers a single formula $\phi$ which is either satisfied or not. In contrast, question restrictions define separate requirements for both *acceptance* and *rejection*, the latter of which discards paths without affecting the result.

There are some model checking approaches which address time or probability or both [44], [47]; a timed form of Petri net can also be verified [32]. However, these approaches are still subject to the fundamental differences between question restrictions and model checking listed above. The method proposed in [17] for GSPNs goes further by computing the probability of a path in the associated continuous-time Markov chain with vanishing states (VCTMC) satisfying a continuous stochastic logic (CSL) formula. This approach improves on the result provided, but CSL is CTMC-specific and still relies on a single formula rather than separate acceptance and rejection conditions.
Chapter 7

Restricted Directed Acyclic Graphs

Although choice-point nets, augmented reachability graphs and question restrictions offer the means to create a model and answer questions about it, the approach as a whole suffers from a significant problem. Due to the tremendous number of possible event sequences that may be generated during an emergency, the augmented reachability graph associated with a choice-point net model may suffer from insurmountable state-space explosion. Precomputing all possible system behaviour into a single graph for analysis may not be computationally feasible, let alone desirable.

Restricted directed acyclic graphs are introduced in this chapter to reduce the effects of state-space explosion. These structures are inspired by limited lookahead trees [20], [66] in the field of discrete-event systems. One of these trees contains a portion of the system behaviour calculated from the current state. Similarly, rather than unravelling the full and complete behaviour of a choice-point net $\mathcal{N}_{CP}$ into an augmented reachability graph $\mathcal{RG}_{CP}$, which may be prohibitively large, only the paths
in $\mathcal{RG}_{CP}$ that satisfy a set of question restrictions $QR$ are explored. These paths are combined into a directed acyclic graph that is restricted by $QR$, hence the name given to these graphs. In this manner, questions may be answered by individually calculating portions of $\mathcal{RG}_{CP}$ on demand instead of requiring precomputation.

A restricted directed acyclic graph (RDAG) $QRS_{CP} = (\Lambda, Y, \eta, y_0)$ has a similar set of elements to those in an augmented reachability graph: an alphabet $\Lambda \subseteq T^+ \times C^+ \times [0, 1] \times \mathbb{N}$, a set of states $Y \subset \mathbb{N}^{P} \times \chi \times QR \times QR \times \mathbb{N}$, an edge function $\eta: Y \times \Lambda \rightarrow Y$ and an initial state $y_0 \in Y$. However, there are some differences between the two. First, nodes within $QRS_{CP}$ have a cumulative time value $\tau$, rather than a global clock value $\theta$ as in $\mathcal{RG}_{CP}$, and contain acceptance ($A$) and rejection ($J$) question restriction sets. Second, edges represent sequences of event firings and choice selections. The alphabet is defined accordingly; the notation $T^+$ and $C^+$ represents the sets of all nonzero concatenations of transitions and choices, respectively. Third, the firing numbers $\rho$ located within the nodes of $\mathcal{RG}_{CP}$ are not necessary within $QRS_{CP}$.

Restricted directed acyclic graphs contain fewer nodes than augmented reachability graphs because they do not store intermediate conflict nodes. The possibility of several events occurring at the same time is represented by a series of possible paths in $\mathcal{RG}_{CP}$. These paths contain many intermediate conflict nodes that represent states in which the system spends virtually no time. Although these paths provide a precise description of what may occur during a conflict, such a description is not entirely necessary, especially at the expense of state-space reduction. Restricted directed acyclic graphs contain collapsed versions of these paths on single edges between the start node and end node. The sequence of events is still preserved in that these composite
edges now represent any number of transition occurrences and choice selections. This permits the removal of the firing numbers \( \rho \) present in the nodes of \( \mathcal{RG}_{CP} \) as their purpose is to track conflict which is resolved in \( \mathcal{QRG}_{CP} \).

For example, consider one of the questions posed for the running example: what is the probability that both residents’ illness will be discovered within 24 hours? The question restrictions associated with this question are contained in \( A = \{ (\text{Isolate, geq, 2}) \} \) and \( J = \{ (\text{time, gt, 24}) \} \). The restricted directed acyclic graph corresponding to these
sets is shown in Figure 7.1. The second question concerns the likelihood that both residents are still undiscovered after 48 hours, resulting in $A = \{(\text{rIll, eq}, 2), (\text{time, geq}, 48)\}$ and $J = \{(\text{Isolate, gt}, 0)\}$. The restricted directed acyclic graph corresponding to this second question is shown in Figure 7.2.

Figure 7.2: Restricted Directed Acyclic Graph Generated From $A = \{(\text{rIll, eq}, 2), (\text{time, geq}, 48)\}$ and $J = \{(\text{Isolate, gt}, 0)\}$ for the LTCH Protocol

Both of these graphs demonstrate the advantages of using restricted directed acyclic graphs: the results are far more compact, readable and require far less computation. Indeed, the only path that is actually explored in the process of creating the graph shown in Figure 7.2 is the result itself.
7.1 Conflict Directed Acyclic Graphs

Conflict directed acyclic graphs (CDAGs) are temporary structures built during the construction of a restricted directed acyclic graph. A conflict directed acyclic graph $\mathcal{CG}_{CP}$ initially represents a portion of an augmented reachability graph $\mathcal{RG}_{CP}$ dedicated to the possible outcomes from a series of events that may occur at the same time. This graph is then resolved, a process whereby its edges are merged to represent sequences of events and choices. These amalgamated edges are then appended to a restricted directed acyclic graph $\mathcal{QRG}_{CP}$.

As a result, a conflict directed acyclic graph $\mathcal{CG}_{CP} = (\Lambda, Z, \kappa, z_0)$ contains a similar set of elements to those in both $\mathcal{RG}_{CP}$ and $\mathcal{QRG}_{CP}$: an alphabet $\Lambda \subseteq T^+ \times C^+ \times [0,1] \times \mathbb{N}$, a set of states $Z \subseteq \mathbb{N}^{|P|} \times \chi \times \mathbb{N} \times \mathbb{N}$, an edge function $\kappa : Z \times \Lambda \rightarrow Z$ and an initial state $z_0 \in Z$. The alphabet for the edges and the total time value $\tau$ are defined identically to those within $\mathcal{QRG}_{CP}$. However, the firing numbers $\rho$ contained within the nodes are identical to those within $\mathcal{RG}_{CP}$. The sole purpose of a conflict directed acyclic graph is to be resolved. This is accomplished by merging the edges of the graph together to represent sequences rather than individual events and choices. In this manner, intermediate conflict nodes are eliminated and the resulting nodes and edges are added to the restricted directed acyclic graph.

Consider the conflict shown in Figure 7.3(a) where the event `sSeeIllResident` may occur twice in near-simultaneous succession. By merging the `Isolate` edges linking the root node to its child and grandchild, a single edge representing the sequence of the events now connects the root to its grandchild as shown in Figure 7.3(b). Continuing in this manner, the root node is directly connected to two of its grandchildren, eliminating the need for the intermediate child node entirely as shown in Figure 7.4(a).
Chapter 7. Restricted Directed Acyclic Graphs

The result of completing this process appears in Figure 7.4(b). Some of the resolved elements of this graph are then themselves merged into the $\mathcal{QRG}_{CP}$ displayed in Figure 7.1.
Chapter 7. Restricted Directed Acyclic Graphs

7.2 Algorithms

The algorithms that generate restricted and conflict directed acyclic graphs bear a striking resemblance to those that produce augmented reachability graphs and analyze
them with question restrictions. Some of the algorithms are virtually identical, which is not surprising given the common elements in the definitions of the three types of graphs. However, there are enough differences that the previously-defined procedures cannot be re-used.

This section introduces the algorithms necessary to create a restricted directed acyclic graph from a choice-point net and a set of question restrictions. Similarities between these methods and their counterparts will be described. Formal proofs with respect to behaviour will not be repeated. Instead, comments on how existing proofs could be modified for the new algorithms will be provided where applicable.

Algorithm 10 is identical to Algorithm 1 with two exceptions. The first is that one of the input parameters is a conflict node and not a standard ARG node. The second is found on line six: the global clock $\theta$ is calculated from the total time $\tau$ and the number of clock units $CU$. Since the result is the same value, Algorithm 10 is equivalent to Algorithm 1 and a slight rephrasing of Propositions 1 and 2 would apply to Algorithm 10.

The situation is even simpler for Algorithms 11 and 2: the only difference is the type of node given as an input parameter. Neither $\theta$ nor $\tau$ are actually used in these algorithms and all other elements within the nodes are identical. Thus, Algorithm 11 is equivalent to Algorithm 2 and barely modified versions of Propositions 3 and 4 would apply to Algorithm 11 as well.

Algorithm 12 has a few additional differences with its counterpart Algorithm 3 beyond the type of node provided as an input parameter. The first is on line 21 where $\theta$ is determined from the total time $\tau$ and the number of clock units $CU$. The second is on line 33 where the total time is determined rather than the next global clock value.
Algorithm 10: ResetConflictGraphTransitionClocks

Input: The choice-point net $N_{CP} = (P, T, I, C, R, SL, ML, S, MD, CU, M, m_0)$
A particular conflict node thisNode $= (m, \chi, \rho, \tau)$

1) enabledTransitions $\leftarrow$ GetEnabledTransitions($N_{CP}$, $m$)

// Update the clocks of timer transitions
2) forall the $t \in T_R$ do
3) if $t \notin$ enabledTransitions then
4) \hspace{1em} $\chi(t) \leftarrow -1$
5) else if $\chi(t) = -1$ then
6) \hspace{1em} $\chi(t) \leftarrow R_N(t, \tau \mod CU)$
7) end
8) end

// Update the clocks of single lapse transitions
9) forall the $t \in T_{SL}$ do
10) if $t \notin$ enabledTransitions then
11) \hspace{1em} $\chi(t) \leftarrow -1$
12) else if $\chi(t) = -1$ then
13) \hspace{1em} $\chi(t) \leftarrow SL(t)$
14) end
15) end

// Update the clocks of multiple lapse transitions
16) forall the $t \in T_{ML}$ do
17) if $t \notin$ enabledTransitions then
18) \hspace{1em} $\chi(t) \leftarrow \emptyset$
19) else
20) \hspace{1em} while $|\chi(t)| < ED(t, m)$ do
21) \hspace{2em} $\chi(t) \leftarrow \chi(t) \cup \{B_{ML}(t)\}$
22) end
23) end
24) end

The last is on line 34 where the next conflict node is created with the next total time instead of a global clock value. Otherwise, the two algorithms are identical. Due to
Algorithm 11: ResetConflictGraphFiringNumbers

Input: The choice-point net $N_{\text{CP}} = (P, T, I, C, R, SL, ML, S, MD, CU, M, m_0)$

A particular conflict node $\text{thisNode} = (m, \chi, \rho, \tau)$

// Determine the set of transitions that will fire next based on clock values
1) $\text{nextTransitionsToFire} \leftarrow \text{FindNextToFire}(N_{\text{CP}}, m, \chi)$

// Go through each one of those transitions
2) forall the $t \in \text{nextTransitionsToFire}$ do
3) if $t \in T_{MD}$ then
   // If $t$ is marking-dependent, assign the number produced by its
   // function using the current marking
   4) $md \leftarrow MD(t)$
   5) $\rho(t) \leftarrow md(m)$
   // If necessary, alter the firing number from impossible to possible
   6) if $\rho(t) < 0$ then
      7) $\rho(t) \leftarrow 0$
   8) else if $\rho(t) > ED(t, m)$ then
      9) $\rho(t) \leftarrow ED(t, m)$
   end
11) else if $t \in T_{ML}$ then
    // If $t$ is multiple lapse, assign the number of minimum lapses
    12) $\rho(t) \leftarrow MLD(\chi(t))$
13) else
    // If $t$ is single lapse, assign the value one
    14) $\rho(t) \leftarrow 1$
15) end
16) end

the different type of node that is created and returned, Proposition 5 requires some
additional translation to apply it to Algorithm 12. Specifically, all of the supporting
lemmas must be altered to include $\tau$ rather than $\theta$. Although extensive, the process
is fairly simple and would result in an equivalent proposition.

Algorithm 13 deviates further from its ARG counterpart Algorithm 4. These
changes reflect the differences between constructing an augmented reachability graph
and a conflict directed acyclic graph. First, edgeTime is no longer calculated at
the beginning of the method. Lines 1-6 of Algorithm 4 do so to determine whether a
conflict is currently underway or complete; in the latter case, the set of transitions that
Chapter 7. Restricted Directed Acyclic Graphs

Algorithm 12: CreateNextConflictGraphNode

Input: The choice-point net $\mathcal{N}_{CP} = (P, T, I, C, R, SL, ML, S, MD, CU, M, m_0)$
- The current conflict node $currentNode = (m, \chi, \rho, \tau)$
- The set of transition(s) that are currently attempting to fire $nextTransToFire$
- The transition that is currently firing $t$
- The choice that is currently selected $c$
- The current edge time $edgeTime$

Output: The new node $nextNode$

// Create the next transition clock and firing values
1) $nextTransitionClocks \leftarrow$ copy of $\chi$
2) $nextFiringNumbers \leftarrow$ copy of $\rho$

// Reduce the numbers for the transition $t$ that is currently firing
3) $nextFiringNumbers(t) \leftarrow nextFiringNumbers(t) - 1$

// Decrement clocks for lapse transitions that are enabled and counting down
4) if $edgeTime > 0$ then
5)     forall the $t' \in T_{SL}$ do
6)         if $nextTransitionClocks(t') \neq -1$ then
7)             $nextTransitionClocks(t') \leftarrow nextTransitionClocks(t') - edgeTime$
8)         end
9)     end
10)    forall the $t' \in T_{ML}$ do
11)       if $nextTransitionClocks(t') \neq \emptyset$ then
12)           forall the $l \in nextTransitionClocks(t')$ do
13)               $l \leftarrow l - edgeTime$
14)           end
15)       end
16)    end
17) end

// If $t$ is multiple lapse, remove a zero lapse from the clock set
18) if $t \in T_{ML}$ then
19)     $nextTransitionClocks(t) \leftarrow nextTransitionClocks(t) - \{0\}$
20) end

// Continued on next page ...

are currently firing is updated. A conflict directed acyclic graph is solely concerned with a single conflict, thus the need to reset $edgeTime$ or $nextTransitionsToFire$ at the end of the conflict is eliminated.

Otherwise, construction of the conflict graph proceeds in an identical manner with
// Continuing from previous page ...

// Create the next marking
21) nextMarking ← m' where (m, χ, ρ, τ mod CU)((t, c, q, k))(m', χ', ρ', τ' mod CU)

// Go through the transitions that are attempting to fire
22) for all the t' ∈ nextTransToFire do
    // Check for previously-ready transitions that have been
    // pre-empted by t and c or for those that are done firing
23)     if ED(t', nextMarking) = 0 or nextFiringNumbers(t') = 0 then
        // Disable the transition’s clock and firing numbers
        // and remove it from the list
24)         if t' ∉ TML then
25)             nextTransitionClocks(t') ← −1
26)         end
27)         nextFiringNumbers(t') ← 0
28)         nextTransToFire ← nextTransToFire − {t'}
29)     else if ED(t', nextMarking) < nextFiringNumbers(t') then
30)         nextFiringNumbers(t') ← ED(t', nextMarking)
31)     end
32) end

33) nextTotalClock ← τ + edgeTime
34) nextNode ← (nextMarking, nextTransitionClocks, nextFiringNumbers, nextTotalClock)
35) if nextTransToFire = ∅ then
    // If no transitions are currently set to fire, reset the clocks
    // and the firing numbers for this node
36)     ResetConflictGraphTransitionClocks(NCP, nextNode)
37)     ResetConflictGraphFiringNumbers(NCP, nextNode)
38) end
39) return nextNode

the exception of line 10/16 in Algorithm 13/4. For conflict graphs, a value of zero is
given for parameter edgeTime in the recursive call. This ensures that the first edge
in a conflict is assigned edgeTime (nonzero) clock ticks while subsequent edges have
no time assigned (just as in ARGs).

These differences mean that Proposition 6 will not translate easily to Algorithm
13. Alternate proofs are necessary to demonstrate that edgeTime and nextTransitionsToFire are updated correctly. Once accomplished, Lemmas 4-2 and 4-3 would be relatively unchanged.
Algorithm 13: ConstructConflictGraphRecursive

- **Input**: The choice-point net $N_{CP} = (P, T, I, C, R, SL, ML, S, MD, CU, M, m_0)$
- The set of transition(s) that are next to fire $\text{nextTransitionsToFire}$
- The current conflict directed acyclic graph $CG_{CP} = (\Lambda, Z, \kappa, z_0)$
- The current node $\text{currentNode} = (m, \chi, \rho, \tau)$
- The current edge time $\text{edgeTime}$

1. **// Fire each transition that is ready**
   1.1. forall the $t \in \text{nextTransitionsToFire}$ do
     1.1.1. // Explore all the choices for this transition
     1.1.2. forall the $c = (l, b, O) \in C_T(t)$ do
       1.1.2.1. // Create the set of transitions to fire for the recursive call
       1.1.2.2. recTransitionsToFire ← copy of nextTransitionsToFire
       1.1.2.3. // Create the child node and add it to the tree
       1.1.2.4. newNode ← CreateNextNode($N_{CP}$, currentNode, recTransitionsToFire, $t$, $c$, edgeTime)
       1.1.2.5. nextProbability ← $b \div b_{\text{nextTransitionsToFire}}$
       1.1.2.6. nextEdge ← ($t$, $c$, nextProbability, edgeTime)
       1.1.2.7. $\kappa$(currentNode, nextEdge) ← newNode
       1.1.2.8. if newNode $\notin Z$ then
         1.1.2.9. Z ← Z $\cup$ {newNode}
       1.1.2.10. ConstructConflictGraphRecursive($N_{CP}$, recTransitionsToFire, $CG_{CP}$, newNode, 0)
     1.1.3. end
   1.1. end
   1.2. end

Algorithm 14 is a completely new addition that has no counterpart in ARG construction. The purpose of this method is to consolidate each path from the initial node to an end node into a single edge, discarding all intermediate nodes in the process. The algorithm itself uses a straightforward iteration:

- Examine each of the initial node’s child nodes (lines 4-17)
- Examine each child node’s own children (line 6-13)
- Merge the two edges linking the initial node to its grandchild (lines 7-12)
- Discard any child nodes that have no incoming edges (lines 14-16)
Algorithm 14: ResolveConflictGraph

Input: The conflict directed acyclic graph $CG_{CP} = (\Lambda, Z, \kappa, z_0)$

1) notComplete ← true
2) while notComplete do
3) notComplete ← false
4) forall the $e \in \Lambda$ where $\kappa(z_0, e) \neq \bot$ do
5) childNode ← $\kappa(z_0, e)$
6) forall the $f \in \Lambda$ where $\kappa($childNode, $f) \neq \bot$ do
7) grandchildNode ← $\kappa($childNode, $f)$
8) notComplete ← true
9) resolvedEdge ← AppendEdge($e$, $f$)
10) $\kappa(z_0, e) ← \bot$
11) $\kappa($childNode, $f) ← \bot$
12) $\kappa(z_0, resolvedEdge) ←$grandchildNode
13) end
14) if forall $z \in Z$ and forall $g \in \Lambda$, $\kappa(z, g) \neq$childNode then
15) \[ Z ← Z - \{childNode\} \]
16) end
17) end
18) end

- Repeat until the only nodes in the graph are the initial node and its children

An example of this procedure in action is shown in Figures 7.3 and 7.4.

Algorithm 15 is a combination of Algorithms 4 and 7, i.e., it recursively constructs a graph that satisfies a set of restrictions. Lines 2-3 mirror the same lines in Algorithm 4; Lemma 4-1 would translate very easily here. The conflict graph for the next set of transitions to fire is constructed and resolved on lines 4-9. The nodes and edges in the conflict graph are then examined in the forall loops on lines 10-36 and 11-35, respectively. If the given node satisfies the rejection set (line 12), it is ignored and the loop performs another iteration (line 13). If the node satisfies the acceptance set (line 20), it is added if it is not already present (lines 21–23) and the edge is
Algorithm 15: ConstructRestrictedGraphRecursive

**Input:** The choice-point net \( \mathcal{N}_{CP} = (P,T,I,C,R,SL,ML,S,MD,CU,M,m_0) \)
- The restricted directed acyclic graph \( \mathcal{QR}_{CP} = (\Lambda,Y,\eta,y_0) \)
- The current restricted node \( \text{currentNode} = (m,\chi,A,J,\tau) \)
- The current firing number function \( \rho \)

**Output:** True if any descendant satisfies \( A \) and false otherwise

1) \( \text{hasAccepted} \leftarrow \text{false} \)

// Determine the next set of transitions that are set to fire
// based on both timing and firing values
2) \( \text{nextTransToFire} \leftarrow \text{FindNextToFire}(\mathcal{N}_{CP},\chi,\tau \mod CU,\rho) \)
3) \( \text{timeElapsed} \leftarrow \text{GetTimeElapsed}(\text{nextTransToFire},\chi,\tau \mod CU) \)

// Construct the conflict graph for the given set of transitions
4) \( z_0 \leftarrow \{(m,\chi,\rho,\tau)\} \)
5) \( Z \leftarrow \{z_0\} \)
6) \( \kappa \leftarrow \text{new transition function with no entries} \)
7) \( \mathcal{CG}_{CP} \leftarrow (\Lambda,Z,\kappa,z_0) \)
8) \( \text{ConstructConflictGraphRecursive}(\mathcal{N}_{CP},\text{nextTransToFire},\mathcal{CG}_{CP},z_0,\text{timeElapsed}) \)

// Resolve the conflict graph into a single root with children
9) \( \text{ResolveConflictGraph}(\mathcal{CG}_{CP}) \)

// Continued on next page ...
// Integrating this resolved conflict into the main graph
forall the \( z' = (m', \chi', \rho', \tau') \in Z \) where \( z' \neq z_0 \) do
forall the \( e \in \Lambda \) where \( \kappa(z_0, e) = z' \) do
if SatisfiesRestricted\((z', e, J, \lor)\) then continue
end
// Alter the restrictions to reflect what occurred
// in this part of the recursion
nextAccept \leftarrow \text{copy of } A
nextReject \leftarrow \text{copy of } J
AdjustForRestricted\((z', e, \text{nextAccept})\)
AdjustForRestricted\((z', e, \text{nextReject})\)

// Create the next node resulting from this path
nextNode \leftarrow (m', \chi', \tau', \text{nextAccept, nextReject})
if SatisfiesRestricted\((z', e, A, \land)\) then
if nextNode \notin Y then
    \( Y \leftarrow Y \cup \{\text{nextNode}\} \)
end
\eta(\text{currentNode, e}) \leftarrow \text{nextNode}
hasAccepted \leftarrow \text{true}
else
    // If the node is not accepted or rejected, explore further
    if ConstructRestrictedGraphRecursive\((N_{CP}, QRG_{CP}, \text{nextNode, } \rho')\) then
        if nextNode \notin Y then
            \( Y \leftarrow Y \cup \{\text{nextNode}\} \)
        end
        \eta(\text{currentNode, e}) \leftarrow \text{nextNode}
        hasAccepted \leftarrow \text{true}
    end
end
end
return hasAccepted

are with respect to how each method manages choices. Since edges in limited directed acyclic graphs may represent more than one event (and hence more than one choice), the number of relevant choices must be determined. This is accomplished by taking the sequence of choices for the edge and projecting away all those but the one associated with a given restriction. The size of the resulting string is used to
Algorithm 16: AdjustForRestricted

Input: A conflict graph node currentNode = (m, χ, ρ, τ)
A conflict graph edge currentEdge = (t_1t_2...t_j, c_1c_2...c_j, q, k)
A set of question restrictions QR

1) forall the qr ∈ QR do
2) if qr ∈ PR ⊆ {prob} × O × [0, 1] where qr = (prob, o, b) then
3) qr ← (prob, o, b ÷ q)
else if qr ∈ TR ⊆ {time} × O × N where qr = (time, o, u) then
4) qr ← (time, o, u - k)
else if qr ∈ CR ⊆ C × O × N where qr = (d, o, i) then
5) choices ← P(c_1c_2...c_j, {d})
6) qr ← (d, o, i - |choices|)
end
end

Algorithm 17: SatisfiesRestricted

Input: A conflict graph node currentNode = (m, χ, ρ, τ)
A conflict graph edge currentEdge = (t_1t_2...t_j, c_1c_2...c_j, q, k)
A set of question restrictions QR
A binary Boolean operator ∇

Output: A Boolean representing whether the current path satisfies QR according to ∇

satisfied ← ⊥
2) forall the qr ∈ QR do
3) if qr ∈ MR ⊆ P × O × N where qr = (p, o, i) then
4) result ← o(M(p), i)
else if qr ∈ PR ⊆ {prob} × O × [0, 1] where qr = (prob, o, b) then
5) result ← o(q, b)
else if qr ∈ TR ⊆ {time} × O × N where qr = (time, o, u) then
6) result ← o(k, u)
else if qr ∈ CR ⊆ C × O × N where qr = (d, o, i) then
7) choices ← P(c_1c_2...c_j, {d})
8) result ← o(|choices|, i)
end
end
13) if satisfied = ⊥ then
14) satisfied ← result
15) else
16) satisfied ← satisfied ∇ result
end
18) return satisfied

adjust the number of remaining values on line 8 of Algorithm 16 and check whether the restriction has been satisfied on line 11 of Algorithm 17.
Finally, Algorithm 18 virtually mirrors Algorithm 5; it begins the recursive construction described in this section by creating an empty RDAG with an initial node and calling Algorithm 15. The first differences between the two methods can be found on line 3, where an RDAG edge function is now defined in place of an ARG edge function. An initial conflict graph node is created on line 4 and is subjected to similar reset procedures on lines 5-6. An initial restricted node is then defined on line 7 and the empty restricted directed acyclic graph is created on line 8. The recursive call is made on line 9, matching line 8 in Algorithm 5.

With this last method, the set of algorithms necessary to produce a restricted directed acyclic graph from a choice-point net and two sets of question restrictions is complete.
7.3 An Example of RDAG Construction

To provide further insight into the inner workings of the algorithms defined in the previous section, the production of part of the graph shown in Figure 7.1 will be examined. This walkthrough will begin with a discovered node and highlight its course through all of the methods but Algorithm 18, which begins the recursive process with question restriction sets $A = \{\text{Isolate, } geq, 2\}$ and $J = \{\text{time, } gt, 24\}$.

**Calling ConstructRestrictedGraphRecursive**

Consider the restricted graph node with marking $m = [2 \ 0 \ 0]$, $\chi(s\text{SeeIllResident}) = 23$, $\chi(r\text{DoneIsolation}) = \emptyset$, $A = \{\text{Isolate, } geq, 2\}$, $J = \{\text{time, } gt, 16\}$ and $\tau = 8$. The rejection restriction $J$ has been adjusted by the 8 hours on the incoming edge from the initial node to the current node. Algorithm 15 is called with this node as `currentNode`, $\rho(s\text{SeeIllResident}) = 2$ and $\rho(r\text{DoneIsolation}) = 0$. Lines 1–3 set `hasAccepted = false`, `nextTransitionsToFire = \{s\text{SeeIllResident}\}` and `timeElapsed = 15`. Lines 4–7 initialize the conflict graph which is constructed on line 8 via a call to Algorithm 13.

**Calling ConstructConflictGraphRecursive**

Within Algorithm 13 the input parameter `currentNode` contains $m = [2 \ 0 \ 0]$, $\chi(s\text{SeeIllResident}) = 23$, $\chi(r\text{DoneIsolation}) = \emptyset$, $\rho(s\text{SeeIllResident}) = 2$, $\rho(r\text{DoneIsolation}) = 0$ and $\tau = 8$. Other input parameter values are `nextTransitionsToFire = \{s\text{SeeIllResident}\}` and `edgeTime = 15`. Taking $t = s\text{SeeIllResident}$ on line 1, $c = (\text{Isolate, } 0.8, O)$ on line 2 and `recNextTransitionsToFire = \{s\text{SeeIllResident}\}` on line 3, Algorithm 12 is called on line 4.
Calling \textit{CreateNextConflictGraphNode}

During the call to Algorithm 12, lines 1–2 copy \( \chi \) and \( \rho \) into \texttt{nextTransitionClocks} and \texttt{nextFiringNumbers}, respectively. Line 3 sets \texttt{nextFiringNumbers(sSeelllResident)} = 1. Since \texttt{edgeTime} = 15, the \texttt{if} statement on lines 4–17 is entered. There are no single lapse transitions in this net, so the \texttt{forall} loop on lines 5–9 is ignored. Transition \texttt{rDoneIsolation} is multiple lapse and is examined within the \texttt{forall} on lines 10–16 but \texttt{nextTransitionClocks(rDoneIsolation)} = \emptyset and thus the \texttt{if} statement on lines 11–14 is not entered. Since \( t = \texttt{sSeelllResident} \) is not a multiple lapse transition, the \texttt{if} statement on lines 18–20 is ignored.

The new marking \([1 1 0]\) is created on line 21 and assigned to \texttt{nextMarking}. The \texttt{forall} loop on lines 22–32 is then entered. For the only iteration, \( t' = \texttt{sSeelllResident} \), \texttt{nextFiringNumbers(sSeelllResident)} = 1 and \( ED(t', \texttt{nextMarking}) = 1 \). Neither of the conditions on lines 23 and 29 are satisfied and the loop is exited.

The new global clock value \( 8 + 15 = 23 \) is assigned to \texttt{nextTotalClock} on line 33; \texttt{nextNode} is created from all of the \texttt{next} variables on line 34. Since \texttt{nextTransitionsToFire} = \{\texttt{sSeelllResident}\} \neq \emptyset, the \texttt{if} statement on lines 35–38 is never entered. This call to Algorithm 12 therefore ends and control is returned to Algorithm 13.

Returning to \textit{ConstructConflictGraphRecursive}

With \texttt{nextNode} complete in Algorithm 13, line 5 sets the probability \texttt{nextProbability} of this choice to \( 0.8 \div 1 = 0.8 \). Line 6 creates the edge \texttt{nextEdge} from \( t, c, \texttt{nextProbability} \) and \texttt{edgeTime} while line 7 makes the entry in the transition function for \texttt{currentNode}, \texttt{nextEdge} and \texttt{nextNode}. At this point, \texttt{nextNode} \( \notin Z \) and so the \texttt{if} statement on lines 8–11 is entered. The node is added to the conflict graph’s set on line 9 and a recursive
call is made on line 10. This call will discover two more nodes (representing the two possible outcomes from another occurrence of `sSeeIllResident`) and attach them to `nextNode` before terminating.

The next iteration of the `forall` loop on lines 2–12 of Algorithm 13 with \( c = (\text{Ignore}, 0.2, O) \) will also yield a second child node and two grandchild nodes, one of which was discovered during the previous iteration. Since `nextTransitionsToFire = \{sSeeIllResident\}`, the single iteration of the `forall` loop on lines 1–13 has now executed and control is returned to Algorithm 15. The completed conflict graph is shown in Figure 7.5.

![Conflict Graph](image)

**Figure 7.5: Walkthrough Conflict Graph After Construction**

**Returning to `ConstructRestrictedGraphRecursive` and Entering `Resolve-ConflictGraph`**

Once the conflict graph has been created on line 8 of Algorithm 15, line 9 is employed to call Algorithm 14 to resolve the graph. The variable `notComplete` is initialized
to true on line 1 and the while loop on lines 2–18 is entered. A value of false is subsequently assigned to notComplete before entering the forall loop on lines 4–17. The first edge examined is $e = (sSIR, Isolate, 0.8, 15)$; line 5 assigns childNode to the node with values $m = [1 \ 1 \ 0]$, $\chi(\text{sSIR}) = 23$, $\chi(\text{rDoneIsolation}) = \{24\}$, $\rho(\text{sSIR}) = 1$, $\rho(\text{rDoneIsolation}) = 0$ and $\tau = 23$. The forall loop on lines 6–13 is entered with $f = (sSIR, Isolate, 0.8, 0)$. Line 7 assigns grandchildNode to the node with values $m = [0 \ 2 \ 0]$, $\chi(\text{sSIR}) = -1$, $\chi(\text{rDoneIsolation}) = \{24, 24\}$, $\rho(\text{sSIR}) = 0$, $\rho(\text{rDoneIsolation}) = 2$ and $\tau = 23$. Since childNode is definitely an intermediate node, notComplete is given a value of true on line 8. The new edge resolvedEdge is constructed from $e$ and $f$ via helper function AppendEdge on line 9 which results in the new edge $(sSIR - sSIR, Isolate - Isolate, 0.64, 15)$. Both $e$ and $f$ are discarded on lines 10–11 while $z_0$ is connected to grandchildNode via resolvedEdge on line 12.

This iteration over lines 6–13 is repeated for $f = (sSIR, Ignore, 0.2, 0)$ and grandchildNode with values $m = [1 \ 1 \ 0]$, $\chi(\text{sSIR}) = 8$, $\chi(\text{rDoneIsolation}) = \{24\}$, $\rho(\text{sSIR}) = 1$, $\rho(\text{rDoneIsolation}) = 0$ and $\tau = 23$. Once complete, all of the edges for childNode have been examined and the loop exits. The if statement on lines 14–16 is then examined. In this case, childNode only had one incoming edge $e$ and that was eliminated on line 10. This satisfies the condition that childNode no longer possesses any incoming edges; the node is removed from the conflict graph’s set on line 15.

The iteration over lines 4–16 continues with $e = (sSIR, Ignore, 0.2, 15)$ and the child node with values $m = [2 \ 0 \ 0]$, $\chi(\text{sSIR}) = 23$, $\chi(\text{rDoneIsolation}) = \emptyset$, $\rho(\text{sSIR}) = 1$, $\rho(\text{rDoneIsolation}) = 0$ and $\tau = 23$. Again, this node’s only
incoming edge is eliminated and the node itself is removed on line 15.

Since notComplete has a value of true, the while loop is entered again. However, during this iteration there are no remaining grandchild nodes; all former grandchild nodes are now children of the initial node. This results in notComplete being assigned a value of false on line 3, but not being reassigned a value of true on line 8 as the forall loop on lines 6–13 is never entered. Thus, the resolution of the conflict graph is completed and control is returned to Algorithm 15. The result of this conflict graph resolution is shown in Figure 7.6.

![Figure 7.6: Walkthrough Conflict Graph After Resolution](image_url)

### Returning to ConstructRestrictedGraphRecursive

Once the conflict graph is resolved, Algorithm 15 examines all the nodes $z'$ in the graph which are not the initial node (as that is already part of the current restricted directed acyclic graph) in the forall loop on lines 10–35. In this case, $z'$ is the node with values $m = [0 \ 2 \ 0]$, $\chi(s\text{See}!\text{ Resident}) = -1$, $\chi(r\text{Don}!\text{Isolation}) = \{24, 24\}$,
\( \rho(\text{Sell Resident}) = 0, \rho(\text{DoNotIsolate}) = 2 \) and \( \tau = 23 \) (the one on the lower left of Figure 7.6). Each edge \( e \) in the conflict graph which links \texttt{currNode} to the conflict node is examined in the \texttt{forall} loop on lines 11–35. The node \( z' \) has one incoming edge \( e = (s\text{SIR} - s\text{SIR}, \text{Isolate} - \text{Isolate}, 0.64, 15) \). It is with these values for \( z' \) and \( e \), as well as \( J = \{(\text{time}, \text{gt}, 16)\} \) and the Boolean operator \( \lor \) that Algorithm 17 is called in the \texttt{if} statement on line 12 to test whether \texttt{nextNode} is on a path that must be rejected.

**Calling \texttt{SatisfiesRestricted}**

Within Algorithm 17, line 1 sets \texttt{satisfied} to a value of \texttt{nil/\perp} to demonstrate that no restriction has yet been tested. Each restriction \( qr \) in \( QR \) is then examined in the \texttt{forall} loop on lines 2–13. Since \( QR \) contains a single element \( qr = (\text{time}, \text{gt}, 16) \), the loop is given a single iteration. The type of the restriction is tested in the \texttt{if} statements on lines 3 (marking), 5 (probability), 7 (time) and 9 (choice), the second-last of which matches \( qr \) thus entering line 8. This line calculates \( o(15, 16) = 15 \text{gt} 16 \), which is \texttt{false} and is assigned to the variable \texttt{result}.

The \texttt{if} statement on line 13 is then entered and since \texttt{satisfied} has a value of \texttt{\perp}, \texttt{satisfied} is given a value of \texttt{false} on line 14. The \texttt{forall} loop ends on line 18, a value of \texttt{false} is returned on line 19 and control is returned to Algorithm 15.

**Returning to \texttt{ConstructRestrictedGraphRecursive} and Calling \texttt{AdjustForRestricted}**

Since the rejection restrictions were not satisfied, Algorithm 15 moves on to lines 15–16 where the acceptance and rejection sets are copied into \texttt{nextAccept} and \texttt{nextReject}. 
Algorithm 16 is then called on lines 17 and 18 to alter these sets based on the current values being explored in the graph.

Algorithm 16 is primarily composed of a **forall** loop that examines each question restriction $qr$ in the set $QR$ (lines 1–10) and tests its type. Marking restrictions are not adjusted as they must stay constant, but probability (line 2), time (line 4) and choice (line 6) restrictions are altered to reflect the event occurrence(s) leading up to the current node. The single choice restriction in **nextAccept** is altered from (Isolate, $geq$, 2) to (Isolate, $geq$, 1) on line 8 since there is one Isolate choice on edge $currentEdge$. The single time restriction in **nextReject** is changed from ($time$, $gt$, 16) to ($time$, $gt$, 1) on line 5 as $currentEdge$ represents 15 hours of elapsed time. A single iteration is all that is required for both of these sets and control is returned to Algorithm 15.

**Returning to ConstructRestrictedGraphRecursive**

The next restricted node **nextNode** is created on line 19 from $z'$ by dropping the firing numbers $\rho$ and adding the question restriction sets **nextAccept** and **nextReject**. On line 20, Algorithm 17 is called again with $A = \{(Isolate, geq, 2)\}$ and the Boolean operator $\land$. In this case, the acceptance conditions are satisfied: the result of $o(|Isolate − Isolate|, 2) = 2 geq 2$ is $true$ on line 11 of that algorithm. This results in the **if** statement on line 20 being entered. Line 21 tests whether **nextNode** is already a part of the current restricted directed acyclic graph; it is not, so the **if** statement is entered and the node is added to the graph’s set on line 22. The transition function is then updated on line 24 with $e$ and, since an acceptable path has been discovered, **hasAccepted** is set to $true$ on line 25. The **else** on line 26 is therefore ignored and
this iteration of the `forall` loop on lines 11–35 ends. As there was only one incoming edge for $z'$, no second iteration occurs and the algorithm returns to line 10.

In the next iteration of the `forall` loop on lines 10–36, $z'$ is assigned to the node with values $m = [1 \ 1 \ 0]$, $\chi(s\text{SeelllResident}) = 8$, $\chi(r\text{Donelsolation}) = \{24\}$, $\rho(s\text{SeelllResident}) = 1$, $\rho(r\text{Donelsolation}) = 0$ and $\tau = 23$ (the one centered at the bottom of Figure 7.6). Each incoming edge $e$ is then examined in the `forall` loop on lines 11–35. The first edge considered is $e = (s\text{SIR} - s\text{SIR}, \text{Isolate} - \text{Ignore}, 0.16, 15)$. Algorithm 17 is called in the `if` statement on line 12; the return value is `false` for the same reason as the previous node. The sets `nextAccept` and `nextReject` are created and updated on lines 15–18; node `nextNode` is created from $z'$ and these sets on line 19. The `if` statement on line 20 calls Algorithm 17; here, the return value is `false` as there is one fewer Isolate choice on $e$ than is required. These cause the `else` statement to be entered on line 26.

**Calling `ConstructRestrictedGraphRecursive` Recursively**

A recursive call is made to this algorithm on line 27 within an `if` statement. If some acceptable path is found, the restricted directed acyclic graph is updated on lines 28–32 in an identical manner to that on lines 21–25. However, no acceptable path is found for $z'$; all outgoing edges have a time of 9 hours, which is greater than the rejection restriction of $(time, gt, 1)$. The node $z'$ and edge $e$ are therefore ignored (and effectively discarded) in this case and this iteration of the inner `forall` loop ends.

There is a second incoming edge for $z'$: $e = (s\text{SIR} - s\text{SIR}, \text{Ignore} - \text{Isolate}, 0.16, 15)$. This edge is processed in an identical manner to its twin as described previously. Thus ends the `forall` loop on lines 11–35 and the algorithm returns to line 10 to examine
the last child node in the conflict graph.
Chapter 8

Modelling a Real-World Protocol

To fully illustrate how choice-point nets may be used to represent emergency response protocols, this section will present a complete model of such a system. The protocol modelled here is “A Guide to the Control of Respiratory Infection Outbreaks in Long-Term Care Homes” from the Ontario Ministry of Health and Long-Term Care [26].

8.1 Assumptions

Certain assumptions have been made to make this example more efficient. These assumptions do not make the model any less realistic, but they do simplify it from a structural perspective.

The LTCH in question is intended for residents whose health is relatively stable (i.e., all chronic conditions are being maintained medically) [55]. The majority of residents are frail [57] and cannot walk without assistance (e.g., a walker or wheelchair) [55]. The LTCH is not large and is contained on a single floor; as such is considered a single geographic area or unit [26, 2.2.5 (i)]. A single unit contains 40 residents
and between three and six staff members depending on the time of day [55], [57]. All registered staff are trained to collect nasopharyngeal specimens [26, 2.1.3]. The LTCH has also conducted its annual general review of policies [26, 2.1.3].

The illness modelled in this example is a non-life-threatening upper-respiratory tract illness [26, 2.2.4 (i)], \textit{i.e.,} a common cold\textsuperscript{1}. The cold shares some symptoms with influenza (malaise and a fever [26, 2.2.4 (i)]) and also causes a dry cough, which combined with a fever indicates a febrile respiratory illness. All tests for influenza come back negative. No antivirals, prophylaxis or immunization clinics are necessary [26, 3.0.6] and no deaths result from the illness. All immunizations received are irrelevant in this case as no immunization prevents the common cold and the exclusion policy [26, 3.0.6] for non-immunized staff is therefore inapplicable.

\section*{8.2 Modelling Notes}

This model does not include an “exposed” phase for the illness—only “susceptible”, “ill” and “recovered”\textsuperscript{2}. A staff member or resident who has recovered is assumed to be temporarily immune to the illness in question. To make the model more efficient, actions which do not impact the spread of infection within the home are not included:

\begin{itemize}
  \item susceptible residents interacting with each other
  \item susceptible staff examining susceptible residents
  \item recovered residents or staff engaging in any activity
\end{itemize}

\textsuperscript{1}The illness is not influenza [26, 2.2.4 (i)], pneumonia [26, 2.2.4 (ii)], a lower-respiratory tract infection [26, 2.2.4 (iii)], SARS [26, 2.2.4 (iv)], metapneumovirus (MPV) [57] or respiratory syncytial virus (RSV) [57].

\textsuperscript{2}This is in keeping with other medical models such as the classic SIR system.
The subscript “NO” attached to a transition name denotes that the firing number for the associated transition is dependent on the fact that no potential or declared outbreak is currently underway in the home. This is represented in the model by the presence of a single token in place \texttt{noOutbreak}. Any firing number generated for these marking-dependent or single-set transitions is multiplied by $M(\texttt{noOutbreak})$, which will either be a one or a zero. This ensures that the transition will only fire if no outbreak is suspected or declared.

The subscript “PDO” attached to a transition name denotes that the firing number for the associated transition is dependent on the fact that a potential or declared outbreak is currently underway in the home. This is represented in the model by the presence of a single token in place \texttt{potentialOutbreak} or in place \texttt{declaredOutbreak}. Any firing number generated for these marking-dependent or single-set transitions is multiplied by $(M(\texttt{potentialOutbreak}) + M(\texttt{declaredOutbreak}))$, which will either be a one or a zero. This ensures that the transition will only fire if an outbreak is suspected or declared.

A single token is shared by places \texttt{noOutbreak}, \texttt{potentialOutbreak} and \texttt{declaredOutbreak}. These three places form a $p$-invariant within the model. With only one token, this ensures that the outbreak situation for the home may only be in one “state” (none, potential or declared) at any one time.

Diagrams for the model are displayed in a modular manner (i.e., one or two transitions per figure) because the model is too large to be illustrated in its entirety. Transitions with marking-dependent firing functions will be illustrated with all of the places included in that function. This may result in “disjoint” or unconnected places in individual figures. However, any such place is only disconnected from the
transition shown; it is connected to other transitions within the model. Choices will be illustrated in these diagrams as they are in Figure 4.1. Specifically, transitions with a single choice will be shown with solid output arcs while transitions with multiple choices will be shown with styled output arcs (dotted, dashed, etc.) where all arcs in the same style belong to the same choice. All marking-dependent firing functions are named $f_#$ or $g_#$ where # represents the number of the figure in which the associated transition appears.

8.3 Common Meals for Residents

Residents of the LTCH share meals together in a large dining room. Breakfast begins at 8AM, lunch begins at 12PM and dinner begins at 5PM. During these meals, ill residents will expose susceptible residents to the cold. Figure 8.1 demonstrates this exposure. Places $r^{\text{IllReported}}$ (shown in Figure 8.7) and $r^{\text{IllUnreported}}$ store tokens for residents who have and have not yet been discovered to be ill by the LTCH.
staff, respectively. These places track resident activities (meals, interactions, etc.) while places $\text{numRI}11\text{LT}72$ and its companion $\text{numRI}11\text{GT}72$ (shown in Figure 8.14) are used to track progression of the cold and recovery. These places store tokens for each ill resident, unreported or reported, who has been ill for less than or more than 72 hours, respectively.

Virtually all residents attend common meals before an outbreak is declared. Those ill residents who have been reported ($r\text{IillReported}$) are not isolated in their rooms and may still take meals with the others [55] but are placed at a separate table [57].

Depending on the number of ill residents, some susceptible residents may not be exposed (not modelled as it does not impact the spread of infection) and some will (transition $r\text{SusceptibleMealExp}$). The number of residents that are exposed is determined using the function $f_{8.1}(m)$ defined using a hypergeometric distribution as follows:

$$f_{8.1}(m) = \left\lceil S \times \left(1 - \frac{(S+R)(IU)}{(S+R+IU)}\right)\right\rceil$$

where $S = M(r\text{Susceptible})$, $R = M(r\text{Recovered})$ and $IU = M(r\text{IillUnreported})$. This function determines the likelihood of a susceptible resident being seated near an ill but unreported resident (three others at the same table and two others at adjacent tables)$^3$, multiplied by the number of susceptible residents and (pessimistically) rounded up. A resident has a 40% chance of resisting the illness; there is a 60% chance that he or she will succumb.

---

$^3$The range for exposure is approximately 2 metres and has the most impact on those facing the infected individual [57] but will also impact those on both sides [56].
It is vital to note that the transition $r\text{SusceptibleMealExp}$ does not require input arcs from the place $r\text{IllUnreported}$ which stores tokens for ill residents who have not been discovered. This would be a structural requirement in many Petri net formalisms, but it is not necessary here for choice-point nets. The firing function $f_{8,1}(m)$ uses the marking values of both $r\text{Susceptible}$ and $r\text{IllUnreported}$ to calculate how many susceptible residents are exposed. Choice-point nets offer a real advantage in this respect as marking-dependent firing functions may use any place’s marking value, not just input places.

### 8.4 Residents Interacting With Others

During the day, residents of the long-term care home will occasionally interact with each other. This occurs during the morning, afternoon and evening outside of meals and group activities\(^4\) (visit, chat, etc.). Figure 8.2 demonstrates this exposure. As in\(^4\) Depending on the residents’ condition, mingling may not occur as frequently as modelled here [57].

---

\(^4\) Depending on the residents’ condition, mingling may not occur as frequently as modelled here [57].
Figure 8.1, no input arcs from any place storing tokens for ill residents are required. The number of residents exposed is managed by $f_{8.2}(m)$, defined as follows:

$$
f_{8.2}(m) = \left[ M(r\text{Susceptible}) \times \frac{M(r\text{IllUnreported})+M(r\text{IllReported})}{M(r\text{Susceptible})+M(r\text{IllUnreported})+M(r\text{IllReported})+M(r\text{Recovered})} \right]
$$

Before an outbreak is suspected or declared, reportedly ill residents are free to mingle with the rest of the home’s population. This scenario will change if an outbreak is declared, as discussed in a future section.

In addition to mingling, there are several group activities or gatherings which occur in the home. These include clustering around the medication cart before breakfast, attending baking group in the afternoon and/or an exercise class in the evening [55]. Figure 8.3 illustrates exposure during these activities. Approximately half of the residents take part in these activities [56] and ill but unreported residents will expose others; as with meals, ill residents may only participate in a separate group. The
number of residents exposed is managed by $f_{8.3}(m)$, defined as follows:

$$f_{8.3}(m) = \left\lceil \frac{S}{2} \times \left(1 - \frac{(S+R)(4-3)}{(S+R+IU)} \right) \right\rceil$$

where $S = M(r\text{Susceptible})$, $R = M(r\text{Recovered})$ and $IU = M(r\text{IllUnreported})$.

This function produces fewer exposed residents than at group meals as only half of residents participate and they are not seated or spaced in the same manner; sufficient contact is assumed to occur with four others, not five.

Susceptible residents will also receive visitors during the day; it is rare for a visitor to be unwell as most are cognizant of the risk of infection for elderly residents [38]. It is in fact so rare that estimates put the number of such visits overall at perhaps one per day [55]. Combined with the probability that an ill visitor will visit a susceptible resident, the impact of ill visitors is considered minimal in this example and will not be modelled.

The protocol gives requirements for non-staff surveillance [26, 2.2.2 (ii)] to prevent visits from ill individuals. This includes activities such as posting signs, providing handwashing facilities and adding reminders to the home’s voicemail system; ill visitors are not asked to leave. Although these actions reduce the number of ill visitors, some may feel that their visit is necessary, choose to ignore the signs, or not yet know that they are sick.
8.5 Staff Examining Residents

Health care providers interact with residents every hour that they are awake (shorter “check-ins” occur overnight), including group meals and group activities [56]. Staff members monitor the residents’ condition and make notes on daily surveillance forms [26, 2.1.3, 2.2.2 (i), 2.2.3 (i)]. They become more attentive to respiratory symptoms as a part of this passive surveillance [26, 2.2.2 (i), 2.2.3 (i)] during influenza season. Staff members and residents spend more than enough time during these interactions to infect each other, especially from hand-to-hand contact and close contact with those who are hard of hearing [55].

Residents who become ill may infect susceptible staff members during these interactions; Figure 8.4 illustrates this event. If the health care provider is infected, a token is shifted from $s_{Susceptible\text{OnDuty}}$ to $s_{Ill\text{OnDuty}}$ and a new token is deposited in

Figure 8.4: Susceptible Staff Members Examining Unreported Ill Residents (No Potential or Declared Outbreak)
numSIllLT72 to track the length of time this care worker has been ill. Regardless of whether the staff member succumbs to the cold or not, a token is deposited in numRIllUnreportedSeen as a health care provider has seen an ill, yet unreported, resident. Tokens in this place will be eliminated later when surveillance data are examined (shown in Figure 8.8).

There are no input arcs from rIllUnreported into transition sSusceptibleSeeR-IllUnreported as these tokens would be simply deposited back into rIllUnreported. However, the marking value of this place is vital to the number of times the transition may fire:

\[
f_{8.4}(m) = 
\left\lfloor M(r\text{IllUnreported}) \times \frac{M(s\text{SusceptibleOnDuty})}{M(s\text{SusceptibleOnDuty})+M(s\text{IllOnDuty})+M(s\text{RecoveredOnDuty})}\right\rfloor
\]

The function \(f_{8.4}\) multiplies the number of ill and unreported residents by the probability that the staff member is susceptible to determine the number of interactions that match this situation.

Staff members in the long-term care home should be healthy to prevent the spread of illness among the elderly population. It is the policy of the LTCH that ill health care providers must report their symptoms to their supervisor and not go on duty until they are symptom-free. However, ill health care providers may be on duty for a number of reasons not limited to the following:

- a staff member may only develop symptoms during a shift and decide to finish
- an LTCH may have punitive policies in place (e.g., reduced pay) that a staff member wishes to avoid
- an outbreak could have infected the staff to the point where there are not enough
susceptible staff members to fill the shifts

Ill health care providers may also examine ill residents as shown in Figure 8.5. Although infection is not spread in this case, ill health care providers must still note

\[
\begin{align*}
R : \{7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21\}
\end{align*}
\]

Figure 8.5: Ill Staff Members Examining Unreported Ill Residents

the ill residents’ symptoms on the daily surveillance forms \(\text{numR}1\text{l}1\text{Unreported}\text{Seen}\). The transition \(\text{sIllSeeR}1\text{l}1\text{Unreported}\) will fire once for each ill and unreported resident who has not seen a susceptible or recovered staff member:

\[
f_{8.5}(m) = \left[ M(r1\text{l}1\text{Unreported}) \times \frac{M(s1\text{l}1\text{OnDuty})}{M(sSusceptibleOnDuty) + M(s1\text{l}1\text{OnDuty}) + M(sRecoveredOnDuty)} \right]
\]

There is only one “choice” for this scenario; it is represented using solid output arcs. The scenario where a recovered staff member examines an ill resident is identical provided all instances of \(s1\text{l}1\text{OnDuty}\) are replaced with \(s\text{RecoveredOnDuty}\). As a result, it is not shown here.

Ill health care providers may also infect the susceptible residents that they examine, as illustrated in Figure 8.6. Here, tokens are transferred into \(r1\text{l}1\text{Unreported}\) and
Figure 8.6: Ill Staff Members Examining Susceptible Residents (No Potential or Declared Outbreak)

Residents who have been reported as ill will be seen by health care providers as usual before an outbreak is declared. Figure 8.7 illustrates this scenario. The resi-
dent’s condition is still documented by the health care provider on the daily surveillance form but it has no real impact as the resident is already known to be ill. The primary concern in this case is the spread of the infection. The transition $s_{SusceptibleSeeRIllReported}$ fires according to the following function:

$$f_{8.7}(m) = \left[ M(rIllReported) \times \frac{M(sSusceptibleOnDuty)}{M(sSusceptibleOnDuty) + M(sI11OnDuty) + M(sRecoveredOnDuty)} \right]$$

The choice $sResist$ for this transition has a probability of 0.9 because staff are expected to wear masks and gloves when examining residents who have been reported to be ill whether an outbreak has been identified or not [57]. However, compliance is not guaranteed and the $sSuccumb$ choice has a probability of 0.1. The scenario where ill or recovered staff members see residents who have been reported as ill is also not modelled as it does not impact the system’s marking.
A health care provider may fail to notice or report a resident’s symptoms for any number of reasons

- ill residents may be asymptomatic or may attempt to hide their symptoms
- symptoms may not manifest during the examination (e.g., no sneezing)
- symptoms may resemble those related to a pre-existing condition (e.g., some heart conditions lead to coughing)
- symptoms may not appear to be serious
- a health care provider may be distracted or extremely busy and simply forget

All staff members are aware of the LTCH’s policy on noting residents’ symptoms during routine daily care. Entries are made on daily surveillance forms which are then submitted to the Infection Control Professional\(^5\) (ICP) upon the conclusion of hourly examinations. The ICP uses the data on these forms to establish standard levels of infection throughout the year and determine whether the current levels of infection in the home indicate a possible outbreak. Figure 8.8 demonstrates this process. Each token in numRI11UnreportedSeen represents surveillance data about an ill resident who has not yet been discovered. Upon the conclusion of hourly examinations, entries are made on the forms by staff members. These reports must wait for the ICP to check them (shown in Figure 8.19) to determine the status of an outbreak. However, a health care provider will have either correctly identified and noted an ill resident or not. This permits staff members to implement early infection control measures (e.g., using gloves during examinations, separating ill residents at group meals) ahead of confirmation from the ICP.

---

\(^5\)Infection Prevention and Control (IPAC) is a more recent title that is currently in use [57].
In addition to the passive surveillance described in this section, active surveillance is also conducted by the ICP. The protocol states that this “involves seeking out residents with an infectious process” [26, 2.2.3 (ii)]. In this case, the ICP will speak to other staff members about their observations and see residents to confirm the presence of reported symptoms. The administration of the LTCH believes that this approach satisfies the protocol’s suggestion that “the method used by each home should be practical in that setting” [26, 2.2.2 (ii)].

All of the scenarios described in this section will differ somewhat once an outbreak is suspected or declared. The resulting changes will be described in Sections 8.10 and 8.11.


8.6 Staff On and Off-Duty

A health care provider may be in one of the following states due to the spread of infection at the LTCH:

- susceptible and on or off-duty
- ill, but not reported as such, and on or off-duty
- reported as ill and on or off-duty
- recovered and on or off-duty

Shifts for health care providers last eight hours and each shift at the LTCH contains a specific number of staff representing a combination of personal support workers, registered professional nurses and registered nurses. There are six staff members on duty during the day (7AM-3PM), five during the evenings (3PM-11PM) and three overnight (11PM-7AM)\(^6\). The LTCH’s staffing contingency plan [26, 2.1.3] permits ill staff members to work if there are not enough staff members who are not ill available to ensure adequate care for residents. Ill health care providers who have not reported their symptoms will be treated like staff members who are not ill and will be called upon to work at the same rate. If this number is not enough to meet minimum staffing levels then health care providers who have been reported as ill will be asked to work.

Figure 8.9 shows susceptible staff members going on and off-duty during the day. There are no choices for these transitions; tokens are simply shifted from on-duty places to off-duty places and vice-versa. The number of susceptible staff members

\(^6\)These values do not mirror staffing levels at any particular facility but they are similar [56], [57].
Figure 8.9: Susceptible Staff Members Going On And Off Duty During the Day

that work each day shift is defined as

$$f_{8.9}(m) = \left[ 6 \times \frac{M(sSusceptibleOffDuty)}{M(sSusceptibleOffDuty) + M(sIllUnreportedOffDuty) + M(sRecoveredOffDuty)} \right]$$

This function takes the number of health care providers who are supposed to work the
day shift and multiplies it by the probability that an off-duty staff member considered
available to work is susceptible. The scenario for recovered health care providers is
nearly identical (i.e., replace $M(sSusceptibleOffDuty)$ with $M(sRecoveredOffDuty)$)
and is not illustrated here.

Ill health care providers who do not report their symptoms will attempt to work
shifts as any susceptible staff member would; Figure 8.10 demonstrates this during
the day shift. The number of ill (but unreported) health care providers that work the
day shift is defined as

$$f_{8.10}(m) = \left[ 6 \times \frac{M(sIllUnreportedOffDuty)}{M(sSusceptibleOffDuty) + M(sIllUnreportedOffDuty) + M(sRecoveredOffDuty)} \right]$$
The number of these health care providers that end the day shift is defined as

\[ g_{8.10}(m) = M(s\text{I}ll\text{O}n\text{D}uty) - M(\text{numSIllReportedOnDuty}) \]

There may not be enough health care providers who are not reported to be ill to go on shift if the illness has infected a significant proportion of the staff. If this is the case, 
\[ M(s\text{I}ll\text{UnreportedOffDuty}) + M(s\text{SusceptibleOffDuty}) + M(s\text{RecoveredOffDuty}) < 6 \]

The deficit must be made up by asking health care providers who have been reported ill to work\(^7\). Figure 8.11 demonstrates this occurrence. The transition s\text{I}ll\text{ReportedGoOnShift} fires according to the following function:

\[ f_{8.11}(m) = 6 - M(s\text{SusceptibleOffDuty}) - M(s\text{I}ll\text{UnreportedOffDuty}) - M(s\text{RecoveredOffDuty}) \]

\(^7\)Other options available to LTCHs include asking employees to work double shifts, asking high-level employees to fill in for lower-level positions and, in a crisis, using an outside staffing agency [57].
This function will result in a negative value if there are enough susceptible, unreported or recovered staff members available to fill the shift. This negative value will be interpreted by the net as zero, resulting in the transition not firing.

Nearly identical structures to the previous exist to manage the evening and night shifts; they only differ by the times for going on and off-duty and by the number of staff members on shift.

The place `numSIllReportedOnDuty` is used to track how many reportedly ill health care providers are currently working. A separate place could have been created for these staff members on duty but this would have resulted in several additional transitions for health care provider interaction with residents and more complex functions describing their firing numbers. The tradeoff is that modelling other events becomes more complex, but overall the model is more efficient with a single place for ill health care providers on duty.

Ill staff members that have been reported go off-shift as well. Figure 8.12 illustrates how the temporary tokens created when the ill health care providers that have been
Figure 8.12: Reported Ill Staff Members Going Off Duty

reported went on shift are destroyed at the end of the shift.

8.7 Ill Staff Monitoring

Health care providers must assess their own health and any symptoms they experience; this will typically occur twice per shift [57]. Once they are certain they are ill, staff members must report their symptoms to their supervisor [26, 2.2.2 (ii)]. A health care provider will typically work through the remaining portion of the shift rather than call on another staff member to replace him or her mid-shift\(^8\). Symptoms must be extremely severe (such as those associated with influenza) for a staff member to go home early.

Figure 8.13 illustrates this scenario. The number of ill (but unreported) health care providers that assess their own symptoms is calculated using the function

\[
f_{8.13}(m) = M(s\text{I}llOnDuty) - M(\text{numSIllReportedOnDuty})
\]

\(^8\)Mid-shift replacement is unlikely if a health care provider leaves early [57].
Tokens are deposited in \texttt{numSillReportedOnDuty} and \texttt{numSillReports} if an ill staff member is reported. The former ensures that the token representing the health care provider will be transferred to \texttt{sillReported} when the shift ends. The latter registers a report with the supervisor which may be forwarded to the ICP (shown in Figure 8.18).

### 8.8 Recovery From Illness

Residents infected with the cold will gradually recover. It takes approximately three days\textsuperscript{9} for a resident to improve and no longer be contagious; it may take much longer for a resident to return to his or her previous level of health. Figure 8.14 illustrates how ill residents recover in the model. Upon infection, a token is placed in \texttt{numRIillLT72}

\textsuperscript{9}The length of time varies depending on the illness and the resident; it may be 2-5 days before a resident is no longer symptomatic and infectious [56], [57].
Figure 8.14: Ill Residents Recovering From the Cold (Part 1)

and a multiple-lapse timer is set for three days (72 hours). Once the timer has elapsed, the resident has either recovered or is still ill. In the former case, the token is shifted into \texttt{numRIllRecovered} to await moving a token that represents a resident. In the latter case, the token is placed in \texttt{numRIllGT72} to wait for another 24 hours and see if the resident’s condition has improved.

Once it has been determined that a resident has improved, a token must move from one of the “ill” places (\texttt{rIllUnreported} or \texttt{rIllReported}) to \texttt{rRecovered}. A resident who has been reported ill is more likely to recover than an unreported resident. Residents who have been reported ill are virtually guaranteed to have been discovered within 72 hours and will receive some additional care for the cold. However, some ill residents may not have been discovered in that time but did recover on their own.

Figure 8.15 shows how residents move between their permitted “states” in the
Figure 8.15: Ill Residents Recovering From the Cold (Part 2)

model. Once a resident is known to have recovered, tokens are moved from $r_{IllReported}$ to $r_{Recovered}$ as soon as possible (within one clock tick). If there are not enough reported residents, tokens from $r_{IllUnreported}$ are also moved. If a resident is occupied elsewhere, the transitions $r_{IllReportedRecover}$ and $r_{IllUnreportedRecover}$ “wait” for tokens to be returned and then fire once they are enabled.

The recovery scenario is virtually identical for staff members, as shown in Figure 8.16. The primary difference between residents and health care providers recovering is the probability of recovery within three days: it is significantly higher for staff members who are in better health than residents. Otherwise, this structure is identical to that in Figure 8.14.

Similarities also exist where shifting ill health care provider tokens are concerned; Figure 8.17 illustrates this scenario. This figure is nearly identical to that of Figure
Figure 8.16: Staff Members Recovering From the Cold (Part 1)

Figure 8.17: Staff Members Recovering From the Cold (Part 2)
8.15 with the exception of `numSReportedRecovered`. This place stores a token for each ill staff member who was reported as such and recovered. This information will be used in conjunction with the number of reports in `numSillReports` (shown in Figure 8.13) to determine whether some health care providers have been ill long enough to warrant a report to the ICP (shown in Figure 8.18).

### 8.9 Determining a Potential Outbreak

An Infection Control Professional is an employee of the LTCH who is primarily responsible for monitoring illness in the home and determining when an outbreak is occurring [26, 2.2.1 (iv)]. He or she often has other employment duties at the home and is not solely hired for the position\(^\text{10}\). The responsibilities assigned to the ICP include:

- analyzing surveillance data for “sentinel events and trends” [26, 2.2.2 (i)]
- collecting reports of residents’ symptoms via passive surveillance from daily surveillance forms
- collecting reports of staff infection from supervisors
- establishing “baseline levels of infection throughout the year” [26, 2.2.2 (i)] (usually quarterly [57])
- managing non-staff surveillance; this includes activities such as posting signs reminding visitors of infection control measures and ensuring that handwashing facilities or hand hygiene products are available [26, 2.2.2 (iii)]

\(^{10}\)ICPs may be allocated less than one hour per day for the task, depending on the facility [57].
monitoring the medications taken by the residents, especially antibiotics [57].

In this LTCH, the ICP is a staff member who is allocated eight hours per week to perform this analysis [56]. The ICP uses some of this time every day at 8AM while residents are at breakfast. At this point the ICP reviews the submitted surveillance data, compares it to baseline levels of infection and determines whether an outbreak may exist [26, 2.2.3 (iii)].

Infection Control Professionals have two primary sources of passive surveillance data regarding illness in the LTCH:

- Resident data from shift reports (shown in Figure 8.8)
- Staff data from supervisor reports

A supervisor must file a report with the ICP regarding “cases/clusters of employees/contract staff who are absent from work for 72 hours with febrile respiratory infection” [26, 2.2.2 (ii)]. Since the cold modelled in this scenario can cause a dry cough and an elevated temperature, ill staff members will have indicative symptoms and reports must be filed. Figure 8.18 demonstrates how supervisors decide to report cases to the ICP. First, 72 hours must expire for each report of a health care provider falling ill\(^\text{11}\) (numS11Reports). One report is removed for each of these staff members who have recovered (numSReportedRecovered). The supervisor then submits reports to the ICP; one for each health care provider who is still ill more than 72 hours after first reporting symptoms. There is a 95\% chance that a report is actually submitted as the supervisor may forget or possibly ignore it, especially prior to an outbreak.

\(^{11}\)In some LTCHs, staff members who phone in sick contact the scheduling clerk, not a supervisor. If the employee is absent for more than three days then he or she must present a sick note [57]. By this time it is assumed that the supervisor is aware of the employee falling ill.
Figure 8.18: Supervisors Reporting Ill Staff Member Cases to the ICP (No Potential or Declared Outbreak)

The choice representing the 5% chance that a report is forgotten or ignored is not shown as it has no output arcs or places since the token representing a report is destroyed if the report is not submitted. The probability of these choices will change if an outbreak is suspected or declared (shown in Sections 8.10 and 8.11).

The protocol states the following:

Whenever there are two cases of acute respiratory tract illness within 48 hours on one unit, an outbreak should be suspected and tests should be done to determine the causative organism.

The LTCH is contained entirely on one floor and is considered a single geographic unit. However, in reality no outbreak is suspected until there are three or four cases; a
threshold of two cases would cause the LTCH to be in outbreak status the majority of the time [56]. Therefore, any three reports within 48 hours of each other will prompt the ICP to suspect an outbreak [57]. Figure 8.19 demonstrates how three reports are necessary to fire the transition threeReportsIn48 and how reports that are 48 hours old are discarded via illReport48Old. It is important to note that some of the arcs to and from numIllReportsNO and threeReportsIn48 have two arrowheads. Each of these arcs actually represents two individual arcs; they have been collapsed in the diagram for space reasons.

### 8.10 Managing a Potential Outbreak

Once a potential outbreak has been identified, the first task of the ICP is to assess the situation [26, 3.0 (1)]. The ICP must create the line listing\(^{12}\) which will be used (even in a partially completed form) to confirm the outbreak with the local Health

---

\(^{12}\)The line listing is typically updated every day [58].
Unit if necessary. A potential outbreak also requires the immediate implementation of general infection control measures [26, 3.0 (2)]. These measures include:

- reiterating “the need for good hygiene before and after providing care to residents” [26, 3.0 (2)]
- using personal protective equipment (PPE) such as masks and gloves when interacting with ill residents (whereas gowns and eye shields would not be used for an illness with cold-like symptoms) [56]
- isolating ill residents to their rooms

This has a swift and significant impact on many of the day-to-day activities of the LTCH. From a modelling perspective, the vast majority of the changes are not structural and can be briefly described for each figure shown thus far:

8.1 The common meals for residents are not altered; ill residents who have been reported take their meals in their rooms, rather than at separate tables.

8.2 Ill residents who have been reported are isolated and do not interact with other residents during the day. This is represented by a duplicate case with renamed transition \( r_{SusceptibleInteractRill} \) and an identical marking-dependent function save any inclusion of \( M(r_{IllReported}) \).

8.3 Group gatherings for residents are not altered; ill residents who have been reported stay in their rooms and do not take part.

8.4 Health care providers will be more likely to wash their hands following the examinations of all residents. This is represented by a duplicate case with renamed transition \( s_{SusceptibleSeeRillUnreported} \) with choices \( s_{Resist} \) at 0.7 and \( s_{Succumb} \) at 0.3 due to imperfect compliance on the part of staff.
8.5 The case where ill health care providers interact with ill residents (reported or not) is not altered as the use of PPE has no impact.

8.6 Health care providers who are ill will be more likely to wear gloves and masks themselves during the examinations of susceptible residents. This is represented by a duplicate case with renamed transition $s_{IllSeeR_{Susceptible}}P_{DO}$ with choices $r_{Resist}$ at 0.5 and $r_{Succumb}$ at 0.5 due to imperfect compliance on the part of staff.

8.7 Health care providers already use masks and gloves when examining residents who have been reported as ill, but compliance is more likely. This is represented by a duplicate case with renamed transition $s_{SusceptibleSeeR_{IllReported}}P_{DO}$ with choices $s_{Resist}$ at 0.9 and $s_{Succumb}$ at 0.1.

8.8 Once a potential outbreak is identified or an outbreak is declared, staff will be more vigilant with respect to symptomatic residents. This is represented by a duplicate case with renamed transition $s_{SubmitData}P_{DO}$ with choices $submittedAndNoted$ at 0.98 and $missed$ at 0.02 and renamed place $numIllReportsP_{DO}$.

8.9 Care workers go on and off duty is the same manner regardless of outbreak status; there is no change to this case or those illustrated in Figures 8.10, 8.11 or 8.12.

8.13 Ill care workers are more likely to report their symptoms once an outbreak has been declared [57]. This is represented by a duplicate case with renamed transition $s_{IllOnDutySelfAssess}P_{DO}$ with choices $report$ at 0.95 and $ignore$ at 0.05.
8.14 Both residents and care workers recover from the illness at the same rate regardless of the LTCH’s outbreak status. This case is not altered and the same is true for Figures 8.15, 8.16 and 8.17.

8.18 Reports of ill care workers who are absent are virtually guaranteed to be submitted once a potential outbreak has been identified or an outbreak has been declared. This is represented by a duplicate case with renamed transition $\text{illReportSubmit}_{PDO}$ with choices submit at 0.99 and ignore at 0.01 and renamed place numIllReportsPDO.

Since the cold has several symptoms in common with influenza, tests must be ordered to determine whether the illness is actually influenza\textsuperscript{13}. The protocol also states that “homes are required to call their local public health unit whenever a respiratory outbreak is suspected” [26, 2.2.5 & 3.0 (3)]. In order to send nasopharyngeal swabs for testing the LTCH must alert the local Health Unit that an outbreak is suspected [56]. The Health Unit will issue a potential outbreak number which must accompany the swabs to finance testing [56]. In this scenario these tests will come back negative as influenza is not the culprit.

It is not uncommon for a few residents to not feel well, so the ICP makes the decision to wait for an additional case to develop over the next 24 hours before deciding there is an outbreak and confirming the outbreak with the Health Unit. This sequence of events is shown in Figure 8.20. Here, an additional case must appear within 24 hours to fire transition oneFurtherReportIn24. This transition is also only enabled once an outbreak is considered a possibility (potentialOutbreak).

\textsuperscript{13}“If meet [sic] the criteria for Influenza-like illness, notify Public Health, Lab, Medical Director and Pharmacy to go on Outbreak standby” [26, Appendix 15].
In the event that no more reports are received, the possibility of an outbreak must be eliminated. Figure 8.21 shows how a possible outbreak is dismissed by moving the token from potentialOutbreak to noOutbreak in 48 hours. If a potential outbreak is not “upgraded” to declared, this will disable the “PDO” transitions and re-enable the “NO” transitions.

Although the protocol states that the Medical Officer of Health should declare an outbreak [26, 3.0 (1)], in reality the LTCH liaises with infectious disease and public health nurses and makes the final decision. Declaring an outbreak [26, 3.0
(4)] is actually something of a formality; the ICP typically contacts the Health Unit and confirmation is given after examining the (potentially partially completed) line listing.

### 8.11 Managing and Ending a Declared Outbreak

Once an outbreak has been declared [26, 3.0 (4)], the next step is to notify “individuals associated with the home” (e.g., the administrator, employee health nurse) [26, 3.0 (5)] and hold the outbreak management team’s (OMT) first meeting [26, 3.0 (6)].

The OMT directly oversees the management of all aspects of an outbreak. It should include representatives who have decision-making authority within the home as well as a representative from the health unit.

The first meeting is dedicated to no less than 16 activities, the most relevant to this model being:

- “Review the line listing information to confirm an outbreak exists”
- “Develop a working case definition for the outbreak. A case definition is the criteria that will be used throughout the outbreak to consider a resident or staff member as [sic] outbreak associated case.”
- “Review the control measures necessary to prevent the outbreak spreading”

The OMT takes over many of the tasks associated with the ICP. However, the ICP is typically given the role of outbreak coordinator on the team and “ensures that all decisions of the OMT are carried out, and coordinates all activities required to investigate and contain the outbreak”. The OMT is also responsible for tasks such
as drafting communication documents and confirming a variety of procedures (e.g.,
collecting specimens for testing).

The OMT will continue to monitor the outbreak, collect data and update its
control measures if necessary until the outbreak is over:

Viral respiratory outbreaks can be declared over if no new cases have
occurred in eight days from the onset of symptoms of the last resident
case [26, 3.0 (8)].

Figure 8.22 illustrates this event in the model. The arc from \texttt{daysWithoutNewCase}
to \texttt{declareOutbreakOver} with the curved arrowhead and number “8” represents eight
individual arcs which are collapsed here for space reasons. The transition \texttt{addDayWithoutNewCase} fires every 24 hours according to the function

$$f_{8,22}(m) = 1 - M(\text{numIllReportsPDO})$$

This function ensures that a token is deposited in place \texttt{daysWithoutNewCase} for every day where no report of an ill resident or care worker exists in place \texttt{numIllReportsPDO}. If such report(s) do exist, $f_{8,22}(m)$ will result in a zero or negative value (which will be interpreted by the net as zero).

The transition \texttt{newCaseDiscovered} fires as soon as possible (\textit{i.e.}, with a single lapse of one) according to the function

$$g_{8,22}(m) = M(\text{daysWithoutNewCase}) \times M(\text{numIllReportsPDO})$$

This function ensures that the presence of a report of an ill resident or care worker eliminates every recorded case-free day/token within \texttt{daysWithoutNewCase} as soon as possible. Once eight days have elapsed without a new case, transition \texttt{declareOutbreakOver} fires, shifting the token from \texttt{declaredOutbreak} to \texttt{noOutbreak}.

\section{Analysis Challenges}

While the model described in this chapter accurately captures the spread and containment of a respiratory infection outbreak in a long-term care home, it is unfortunately too large for analysis. Even with the reduction of state-space explosion offered by
restricted directed acyclic graphs, the state space is too large to explore. When executed, the Java implementation of the algorithms in this thesis ended prematurely due to a lack of memory, despite allocating 4GB to the process and reduced the number of people to minuscule levels. However, analysis is feasible on an abridged version of this model and is provided in the next chapter.
Chapter 9

Performing Analysis

Unfortunately, the complete LTCH model proved to be too large for the purposes of analysis. Fortunately, analysis can be performed on an abridged version of this model. This representation is mathematically and structurally similar to its complete counterpart, but reduces event frequency in some cases (e.g., resident group activities), eliminates some events altogether (e.g., staff shift changes) and involves fewer people. The results of the experiments executed using this example offer some interesting and surprising insights into which aspects of the protocol have the most impact on the containment of the infection.

9.1 The Abridged Model

The long-term care home in this example is a small one: there are 10 residents in the facility. The residents are fairly independent: they can typically walk without assistance and any chronic conditions they possess are successfully maintained medically. There is one staff member on duty 24 hours a day; five staff members in total are
employed by the home.

\section*{9.1.1 Resident Interaction}

Group meals occur at 8AM, 12PM and 5PM as shown in Figure 9.1. This scenario is similar to that shown in Figure 8.1, as is the marking-dependent function for transition \texttt{rSusceptibleMealExp}:

\[
f_{9.1}(m) = rS \times \left( 1 - \frac{(rS+rR)(rI)}{(rS+rR+rI)5} \right)
\]

\[
= rS \times \left( 1 - \frac{(rS+rR)(rS+rR-1)...(rS+rR-4)}{(rS+rR+rI-nRRI)(rS+rR+rI-nRRI-1)...(rS+rR+rI-nRRI-4)} \right)
\]

In this function, \(rS = M(\texttt{rSusceptible})\), \(rR = M(\texttt{rRecovered})\), \(rI = M(\texttt{rIll})\) and \(nRRI = M(\texttt{numRillReported})\). The result is rounded to the nearest positive whole number.

Residents also interact with each other, but not extensively throughout the day as many are able to leave the home for other activities. The illustration associated with
the transition $r_{SusceptibleInteractRll}$ is identical to that shown in Figure 9.1, but for $R : \{10, 14\}$ and $MD : f_{interact}(m)$ where

$$f_{interact}(m) = rS \times \frac{rI - nRRI}{rS + rI + rR - nRRI}$$

This event mimics the event illustrated in Figure 8.2 with one exception: residents who are known to be ill are isolated and do not interact with others in the home both before and after an outbreak is declared. This decision was made by the facility’s administration due to the home’s size, as a single infected person can expose much of the population in a matter of hours.

There is typically one group activity in the home every evening at 7PM. Again, the illustration associated with the transition $r_{GroupGathering}$ is identical to that shown in Figure 9.1, but for $R : \{19\}$ and $MD : f_{group}(m)$ where

$$f_{group}(m) = rS \times \left(1 - \frac{(rS + rR)(rS + rR - 1)(rS + rR - 2)(rS + rR - 3)}{(rS + rR + rI - nRRI)(rS + rR + rI - nRRI - 1)(rS + rR + rI - nRRI - 2)(rS + rR + rI - nRRI - 3)}\right)$$

This event’s twin is shown in Figure 8.3, especially with respect to the marking-dependent firing function.

### 9.1.2 Staff Examinations

Since the residents of the home are independent and in reasonable health, staff do not examine them frequently. Health care providers see residents three times a day: before breakfast at 7AM, after lunch at 2PM and before bed at 9PM. Residents are also checked once overnight, but there is not enough interaction to spread infection.

A susceptible staff member may examine an ill but yet unreported resident as
shown in Figure 9.2. This event mimics the event illustrated in Figure 8.4. In this

![Diagram](image)

Figure 9.2: (Abridged) Susceptible Staff Members Examining Unreported Ill Residents

abridged model, reporting does not wait until a staff member has completed his or her rounds; if a health care provider realizes that a resident is ill, the staff member should act immediately. As a result, the transition \( s\text{SusceptibleSeeRillUnreported} \) has four choices representing all possible combinations of the staff member resisting or succumbing to the infection as well as reporting or missing the resident’s symptoms. The marking-dependent function for this transition is defined as

\[
f_{9.2}(m) = (rI - nRRI) \times \frac{sS}{sS + sI + sR - nSRI}
\]

where \( sS = M(s\text{Susceptible}) \), \( sR = M(s\text{Recovered}) \), \( sI = M(s\text{Ill}) \) and \( nSRI = M(\text{numSillReported}) \).

An ill staff member may also examine an ill but yet unreported resident as shown
in Figure 9.3. This event’s inspiration is shown in Figure 8.5. The marking-dependent

\[ f_{9.3}(m) = (rI - nRRI) \times \frac{sI-nSRI}{sS+sI+sR-nSRI} \]

Figure 9.4 illustrates the case where an ill staff members may infect a susceptible resident. This event’s twin is shown in Figure 8.6. The marking-dependent function for transition \texttt{sIllSeeRSusceptible} is defined as follows:

\[ f_{9.4}(m) = rS \times \frac{sI-nSRI}{sS+sI+sR-nSRI} \]

Finally, susceptible health care providers examine residents who have been reported as ill as shown in Figure 9.5. This event mimics the event illustrated in Figure 8.7. The transition \texttt{sSusceptibleSeeRlllReported} fires according to the following function:

\[ f_{9.5}(m) = rI \times \frac{sS}{sS+sI+sR-nSRI} \]
9.1.3 Symptoms and Recovery

Ill staff members who have not yet reported themselves as infected will routinely assess their symptoms, approximately once every four hours. Figure 9.6 illustrates
Figure 9.6: (Abridged) Unreported Staff Members Monitoring Their Symptoms

this scenario, which mimics that shown in Figure 8.13.

Residents and health care providers recover in a different manner than in the full LTCH model. Here, a token is created every day for each resident who was infected that day. Once another day passes, those residents are considered to be recovered.

Figure 9.7: (Abridged) Ill Residents Recovering From the Cold

Using this approach, some residents will be ill for closer to 48 hours while some will only be ill for closer to 24, which varies the recovery rate. This sequence of events is
illustrated in Figure 9.7, which is akin to the events shown in Figures 8.14 and 8.15. The marking-dependent firing functions for the transitions are as follows:

\[
\begin{align*}
    f_{9.7}(m) &= M(r\text{ill}124) - M(r\text{ill}24\text{orLess}) \\
    g_{9.7}(m) &= M(\text{numRill}24\text{orLess}) \\
    h_{9.7}(m) &= M(\text{numRill}24\text{orLess}) - M(\text{numRillReported})
\end{align*}
\]

The situation is similar for staff members, who typically recover within a day. The illustration associated with transitions s\text{ill}12\text{orLess}, s\text{RecoverRep} and s\text{RecoverUnrep} is identical to that shown in Figure 9.7, but for the replacement of all instances of “RIll” with “SIll” and the use of 12-hour lapses, rather than 24.

### 9.1.4 Declaring an Outbreak (Over)

Since only one staff member is on duty at any time, all health care providers are authorized to act as ICPs. As such, they may recognize two reports of illness within 48 hours as an outbreak scenario, as illustrated in Figure 9.8. In this case, a staff

![Figure 9.8: (Abridged) Reports Culminating in a Declared Outbreak](image-url)
member must act quickly; he or she will contact the home’s management as soon
as possible (captured in the model as within four hours) and most likely declare an
outbreak. This sequences of events mimics those shown in Figures 8.19 and 8.20.

When an outbreak is declared, isolation remains in effect for all residents reported
as ill. Staff members will be more likely to recognize or report illness, whether it is
amongst residents or themselves. As such, duplicate examination transitions to those
in Figures 9.2-9.5 exist in this model with adjusted probabilities as in the full LTCH
example.

Once there are no remaining reports of illness amongst residents and staff for 12
hours, the outbreak is declared over. This event is shown in Figure 9.9, which is

\[
\text{declareOutbreakOver} \\
\text{SL} : 12 \\
\text{MD} : f_{9.9}(m)
\]

![Diagram of the outbreak sequence](image)

Figure 9.9: (Abridged) Declaring an Outbreak Over

similar in theme to the events illustrated in Figure 8.22. The marking-dependent
firing function for \text{declareOutbreakOver} is defined as follows:

\[
f_{9.9}(m) = M(\text{outbreak}) \times (1 - M(\text{numRllReported}) - M(\text{numSllReported}))
\]
9.2 Implementation

The algorithms described in this thesis were implemented using the Java programming language. To increase execution time and decrease memory usage, some programmatic additions were made:

- specialized expandable collections were introduced for each element (e.g., nodes, edges) with an efficient searching approach
- where possible, recursion was limited to a certain depth, paused and resumed to encourage garbage collection
- in cases where conflict occurs between transitions that do not steal tokens from each other, the paths are not calculated via a conflict graph but by sequentially determining all firing combinations

9.3 Scenarios and Questions

Virtually all of the experiments described in this section were executed on Intel Core 2 Quad 2.4GHz machines running Ubuntu Linux 10.04.2 (32bit) and Sun Java 6.24 (JDK 1.6.24). 2GB of memory was allocated to the Java Virtual Machine (JVM) in each case. One of the experiments that produced a larger state space required more computing power and memory. It was executed on a 2x Intel Quad-Core Xeon 2.3GHz machine running Ubuntu Linux 10.04.2 (64bit) with the same JVM. 7GB of memory was allocated to the Java Virtual Machine in this case. This experiment will be highlighted in this section with an asterisk.

The administrators of the LTCH described in this chapter want to measure the effectiveness of the strategies which implement the protocol and determine if these
strategies could be improved or relaxed. Specifically, there are questions regarding the need for an aggressive approach to isolation, the importance of health care providers recognizing and reporting symptoms quickly, and whether small alterations to the timing of important events would have a significant impact. These different strategies are captured in four other similar models:

- Residents who are known to be ill are not isolated until an outbreak occurs. This is known as the “less isolation” scenario.

- Staff are less likely to recognize or report residents’ symptoms. Specifically, the probabilities of reporting and missing symptoms pre-outbreak are 0.7 and 0.3, respectively, while the probabilities post-outbreak are 0.9 and 0.1. If they themselves are ill, staff also report and ignore their own symptoms with the same probabilities. This is known as the “less vigilant” scenario.

- Staff may declare an outbreak without consulting management, which results in the transition \texttt{twoReportsin48} being assigned a single lapse of one. This is known as the “faster outbreak” scenario.

- Staff examine residents before known interaction and activity times: examinations now occur at 7AM, 1PM and 6PM. This is known as the “no conflict” scenario.

It should be noted that all of the aforementioned changes were trivial to implement in the model. The “less isolation” scenario required the most work, which simply meant defining a second set of interaction transitions for pre- and and post-outbreak. Making alterations for the remaining scenarios was even less involved. In addition,
scaling the net to include more or fewer people is also trivial: all that needs to be altered is the initial marking.

Three questions about the model were posed:

- What is the probability that at least one half of the residents become ill? This requires accepting all paths that lead to states where the marking of place $r\text{Ill}$ has a value greater than or equal to five. This yields an acceptance set of $A = \{(r\text{Ill}, \geq, 5)\}$.

- What is the probability that more than one half of the staff become ill? This requires accepting all paths that lead to states where the marking of place $s\text{Ill}$ has a value greater than or equal to three. This yields an acceptance set of $A = \{(s\text{Ill}, \geq, 3)\}$.

- What is the probability that an outbreak is declared? This requires accepting all paths that lead to states where the marking of place outbreak has a value equal to one. This yields an acceptance set of $A = \{(\text{outbreak}, \text{eq}, 1)\}$.

These questions were first posed using a time limit of 24 hours, i.e., all paths that were not accepted and exceeded 24 hours were rejected for $J = \{(\text{time}, \text{gt}, 24)\}$. The same questions were then posed again using a probability limit of $10^{-5}$, i.e., all paths that were not accepted and had a cumulative probability less than $10^{-5}$ were rejected for $J = \{(\text{prob}, \text{lt}, 10^{-5})\}$. These produced two different sets of answers relative to a time frame and more likely sequences of events, respectively.
9.4 Results

9.4.1 At Least Half of the Residents Become Infected

The results for the question “What is the probability that at least one half of the residents become ill?” are shown in Table 9.1. Here, by far, the scenario where isolation does not occur until an outbreak is declared produces the worst result. This indicates that the home’s chosen strategy in this regard is excellent. The next largest offender is the case where staff are less vigilant with respect to reporting symptoms. Neither the ability to declare an outbreak faster, nor examining residents before

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Probability</th>
<th>Number of Nodes</th>
<th>Number of Edges</th>
<th>Time to Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.15687</td>
<td>4653</td>
<td>5311</td>
<td>8.29 minutes</td>
</tr>
<tr>
<td>Less Isolation, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.99999</td>
<td>5336</td>
<td>6405</td>
<td>10.55 minutes</td>
</tr>
<tr>
<td>Less Vigilant, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.38318</td>
<td>4653</td>
<td>5281</td>
<td>8.12 minutes</td>
</tr>
<tr>
<td>Faster Outbreak, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.15686</td>
<td>7567</td>
<td>11719</td>
<td>25.23 minutes</td>
</tr>
<tr>
<td>No Conflict, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.16252</td>
<td>5848</td>
<td>5628</td>
<td>11.24 minutes</td>
</tr>
<tr>
<td>Original, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.3119</td>
<td>8685</td>
<td>733</td>
<td>29.22 minutes</td>
</tr>
<tr>
<td>Less Isolation, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.99427</td>
<td>487</td>
<td>72</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Less Vigilant, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.49494</td>
<td>6780</td>
<td>701</td>
<td>18.08 minutes</td>
</tr>
<tr>
<td>Faster Outbreak, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.31015</td>
<td>8870</td>
<td>707</td>
<td>30.34 minutes</td>
</tr>
<tr>
<td>No Conflict, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.30932</td>
<td>7950</td>
<td>701</td>
<td>24.52 minutes</td>
</tr>
</tbody>
</table>

Table 9.1: Analysis: At Least Half of the Residents Become Infected
activities take place, appear to have any real impact.

It is interesting to note that the probabilities resulting from the probability limit are larger than those from the time limit, except for “Less Isolation”. The probability limit allows the graph to be explored further, collecting sequences of events that go beyond 24 hours and have a higher probability of occurring. This is a trend that will be repeated with the results for the remaining questions. Also of note is the increased time and graph size for the time-limited “faster outbreak” results, which is approximately twice the size of its counterparts. This does not appear to be the result of errors in the model, but instead stems from the impact of the approach. Because declaring an outbreak has two choices, and may occur every hour once two reports have been filed, this dramatically increases the number of possible paths.

In addition, the graph size and time required for the probability-limited “less isolation” scenario is quite low. With less isolation, there are more interactions between ill and susceptible residents and staff. This results in transitions firing in near-simultaneous succession more often. The increase in the number of nodes and edges, and therefore paths, is reducing the overall probabilities of those paths. This means that fewer paths in the graph will satisfy the acceptance restriction before the rejection restriction. This highlights the importance of choosing a probability limit low enough to provide meaningful results but high enough to avoid needless computation.

9.4.2 More Than Half of the Staff Become Infected

The results for the question “What is the probability that more than one half of the staff become ill?” are shown in Table 9.2. Here, the same trends continue: less
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Probability</th>
<th>Number of Nodes</th>
<th>Number of Edges</th>
<th>Time to Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original, ( J = {(\text{time}, \text{gt}, 24)} )</td>
<td>0.12790</td>
<td>7542</td>
<td>37437</td>
<td>42.05 minutes</td>
</tr>
<tr>
<td>Less Isolation, ( J = {(\text{time}, \text{gt}, 24)} )</td>
<td>0.80827</td>
<td>8142</td>
<td>46764</td>
<td>53.03 minutes</td>
</tr>
<tr>
<td>Less Vigilant, ( J = {(\text{time}, \text{gt}, 24)} )</td>
<td>0.37304</td>
<td>7542</td>
<td>37347</td>
<td>40.66 minutes</td>
</tr>
<tr>
<td>Faster Outbreak, ( J = {(\text{time}, \text{gt}, 24)} )</td>
<td>0.15686</td>
<td>10794</td>
<td>79330</td>
<td>2.46 hours</td>
</tr>
<tr>
<td>No Conflict, ( J = {(\text{time}, \text{gt}, 24)} )</td>
<td>0.12686</td>
<td>6236</td>
<td>25433</td>
<td>1.29 hours</td>
</tr>
<tr>
<td>Original, ( J = {(\text{prob}, \text{lt}, 10^{-5})} )</td>
<td>0.22708</td>
<td>8651</td>
<td>1171</td>
<td>49.40 minutes</td>
</tr>
<tr>
<td>Less Isolation*, ( J = {(\text{prob}, \text{lt}, 10^{-5})} )</td>
<td>0.45290</td>
<td>11569</td>
<td>918</td>
<td>33.20 hours</td>
</tr>
<tr>
<td>Less Vigilant, ( J = {(\text{prob}, \text{lt}, 10^{-5})} )</td>
<td>0.39024</td>
<td>10747</td>
<td>1656</td>
<td>28.79 minutes</td>
</tr>
<tr>
<td>Faster Outbreak, ( J = {(\text{prob}, \text{lt}, 10^{-5})} )</td>
<td>0.21207</td>
<td>9037</td>
<td>1156</td>
<td>52.68 minutes</td>
</tr>
<tr>
<td>No Conflict, ( J = {(\text{prob}, \text{lt}, 10^{-5})} )</td>
<td>0.23866</td>
<td>8536</td>
<td>1098</td>
<td>47.39 minutes</td>
</tr>
</tbody>
</table>

Table 9.2: Analysis: More Than Half of the Staff Become Infected

Isolation and vigilance prove detrimental, but declaring an outbreak or examining residents earlier has little or no impact. There is overall less probability that staff become infected than residents, likely because staff are better able to resist infection and wear personal protective equipment (PPE) when examining residents known to be ill.

Due to the aforementioned state-space explosion, the results for the “less isolation” scenario had to be computed on a faster machine with more memory. The output from this experiment indicates that during the exploration, conflict graphs allocated space for another 100000 nodes or edges more than 5000 times. Some single conflict
9.4.3 An Outbreak is Declared

The results for the question “What is the probability that an outbreak is declared?” are shown in Table 9.3. There are several items of interest in these results. First,

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Probability</th>
<th>Number of Nodes</th>
<th>Number of Edges</th>
<th>Time to Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.14472</td>
<td>18740</td>
<td>28358</td>
<td>9.78 hours</td>
</tr>
<tr>
<td>Less Isolation, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.95931</td>
<td>20351</td>
<td>37434</td>
<td>13.62 hours</td>
</tr>
<tr>
<td>Less Vigilant, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.34156</td>
<td>18740</td>
<td>27943</td>
<td>9.54 hours</td>
</tr>
<tr>
<td>Faster Outbreak, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.22026</td>
<td>58237</td>
<td>50600</td>
<td>11.39 hours</td>
</tr>
<tr>
<td>No Conflict, ( J = {(\text{time}, \gt, 24)} )</td>
<td>0.18445</td>
<td>26433</td>
<td>28033</td>
<td>4.04 hours</td>
</tr>
<tr>
<td>Original, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.31422</td>
<td>10587</td>
<td>713</td>
<td>36.56 minutes</td>
</tr>
<tr>
<td>Less Isolation, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.56205</td>
<td>22678</td>
<td>797</td>
<td>1.18 hours</td>
</tr>
<tr>
<td>Less Vigilant, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.36186</td>
<td>13159</td>
<td>884</td>
<td>40.08 minutes</td>
</tr>
<tr>
<td>Faster Outbreak, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.41801</td>
<td>12375</td>
<td>724</td>
<td>35.00 minutes</td>
</tr>
<tr>
<td>No Conflict, ( J = {(\text{prob}, \lt, 10^{-5})} )</td>
<td>0.29979</td>
<td>10282</td>
<td>758</td>
<td>36.44 minutes</td>
</tr>
</tbody>
</table>

Table 9.3: Analysis: An Outbreak is Declared

the probability of declaring an outbreak is higher for all probability-limited scenarios except for “less isolation”. This is likely due to the state-space explosion problem for this particular scenario. The increase in the number of nodes and edges, and
therefore paths, is reducing the overall probabilities of those paths. This means that fewer paths in the graph will satisfy the acceptance restriction before the rejection restriction. Second, the faster outbreak scenario increases the likelihood of an outbreak dramatically, but apparently not within 24 hours. Third, a lack of conflict reduces the chance of an outbreak, likely because residents who are identified and isolated sooner do not infect others and produce the conditions for an outbreak to be declared.
Chapter 10

Conclusion

The motivation for this research was to provide a mechanism for modelling and analyzing health care protocols. To that end, this dissertation defines choice-point nets as the framework for modelling, augmented reachability graphs and question restrictions as the tools for analysis and restricted directed acyclic graphs as an efficient alternative to pre-computing the complete reachability graph. Algorithms are provided to convert a choice-point net into an augmented reachability graph, explore the graph with a set of question restrictions, and produce a restricted directed acyclic graph from a choice-point net and a set of question restrictions. A real-world health care protocol is modelled; an abridged version is analyzed using software that implements the aforementioned techniques.

The primary goal of this research was to create a framework that incorporates time and probability, is flexible and scalable, is intuitive for the modeller, represents different decisions and is able to answer the types of questions health care administrators might ask. Choice-point nets represent a successful first step towards satisfying all of these requirements.
Indeed, barring state-space explosion, no caveat regarding the success of choice-point nets would be necessary. Although choice-point nets do not “explode” to the extent other structures (e.g., finite-state automata) do, the mitigation efforts do not go far enough to address real-world protocols without the dedication of vast quantities of computational power and memory. One of the goals of this research was to eliminate the need for a supercomputer to get the job done. That goal was not met.

However, this approach offers two important advantages that are missing from current emergency and disaster modeling mechanisms. First, choice-point nets provide the means to capture failure (likely human) within the protocol. Second, choice-point nets make it possible to determine the likelihood of any result. This is in contrast to existing approaches which tend to estimate a single result.

Although one could argue that state-space explosion becomes less of an issue as computers increase in size and speed, that does not mean that no additional steps should be taken to make choice-point nets more efficient. First, states may be able to be amalgamated within the augmented reachability or restricted direct acyclic graphs. Although the states currently present are unique from a net perspective, it is possible they could be merged based on other criteria. For example, the graph could possess structural properties which graph theory algorithms exploit. The modeller’s question(s) could also render unique nodes equivalent when only a particular subset of behavior is of interest. Second, further steps could be taken to make exploration more efficient in the restricted graph. As it stands, the exploration eliminates some paths but must unravel them to the rejection point first—there is likely some overlap between paths in the rejection process that may be exploited.

There are several features whose addition to choice-point nets would improve their
suitability for health-care modelling. Currently, it is possible to define a marking-dependent firing function for a given transition; a marking-dependent timing function may also prove useful. The function space could also be expanded beyond those from markings to integers: functions could use other properties (e.g., the current time on the global clock) and, taking inspiration from coloured Petri nets, could be defined programmatically with variables, conditionals and loops.

During conflict situations it is considered equally likely that any one event may occur before any other. This is not the case in real life, as some events have priority over others. Choice-point nets would benefit from a mechanism for specifying event priority, as is accomplished in generalized stochastic Petri nets with weights on immediate transitions. Conflict is also an issue for multiple-lapse transitions, which cannot be in conflict with another transition as the theft of a token would adversely affect the multiple lapse timer. A solution to this problem would likely simplify the modelling process for any designer using these types of transitions.

With respect to analysis, acceptance and rejection conditions are presently based upon conjunction and disjunction, respectively. A more flexible approach would permit the use of any Boolean expression using question restrictions for either acceptance or rejection. In addition, while question restrictions and model checking are not identical, they are also not mutually exclusive. Applying model checking techniques to the types of graphs described in this dissertation would provide another tool for phrasing questions and finding answers about a health-care protocol.

In their introductory form, choice-point nets represent a significant departure from the health-care modelling techniques that are currently described in the literature. In particular, the proposed approach to analysis is novel and provides results that do
not appear to have been previously achievable. As a first step, choice-point nets may prove as beneficial as they are tantalizing.
Bibliography


[38] E. Jackson, St. Lawrence Place Retirement Residence. Phone interview, February, 2011.


[58] B. Carter RN GNC(C) OHIV CMMIII, Occupational Health Nurse, County of Frontenac. Personal interview, April 28, 2011.


